FontLab® 4.5

for Macintosh®

User Manual

Copyright ©1992- 2002 by FontLab, Ltd. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher. Any software referred to herein is furnished under license and may only be used or copied in accordance with the terms of such license.

FontLab, *FontLab logo*, *ScanFont*, *TypeTool*, *SigMaker, AsiaFont Studio, FontAudit* and *VectorPaint* are either registered trademarks or trademarks of FontLab, Ltd. in the United States and/or other countries.

Apple, the *Apple Logo*, *Mac*, Mac OS, *Macintosh* and *TrueType* are trademarks of Apple Computer, Inc., registered in the United States and other countries.

Adobe, *PostScript, Type Manager* and *Illustrator* are trademarks of Adobe Systems Incorporated which may be registered in certain jurisdictions.

Windows, *Windows 95*, *Windows 98, Windows XP* and *Windows NT* are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

IBM is a registered trademark of International Business Machines Corporation.

Macromedia, Fontographer and *Freehand* are registered trademarks of Macromedia, Inc.

Other brand or product names are the trademarks or registered trademarks of their respective holders.

THIS PUBLICATION AND THE INFORMATION HEREIN IS FURNISHED AS IS, IS SUBJECT TO CHANGE WITHOUT NOTICE, AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY FONTLAB, LTD.

FONTLAB, LTD. ASSUMES NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR INACCURACIES, MAKES NO WARRANTY OF ANY KIND (EXPRESS, IMPLIED OR STATUTORY) WITH RESPECT TO THIS PUBLICATION, AND EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES OF MERCHANTABILITY, FITNESS FOR PARTICULAR PURPOSES AND NONINFRINGEMENT OF THIRD PARTY RIGHTS.

User manual release 4.50 [11/2002]

Contents

CONTENTS 3

INTRODUCTION 15

About this Manual 16

System Requirements 18

FONTLAB USER INTERFACE 19

Basic Terms 20

Getting Started 24

Customizing FontLab's User Interface 26
- Customization of the Keyboard 27
- Links to External Programs 29

FontLab Windows 31
- Font Window 32
- Glyph Window 34
- Metrics Window 38

Panels 42

EDITING FONTS 45

Opening Fonts 46
- Font Formats 50
- Import Options 51

Creating a New Font 57

Font Window 58

Encoding Modes 63
- Characters, Codes and Glyphs 64
- Names Mode 68
- The Unicode Standard 71
- Unicode Ranges 72
- Codepages 75

Using the Font Window 79
- Navigating 80

Selecting 81
Context Menu 82

Moving Glyphs 85

Saving the Font 87
Autosave 89

Copying and Pasting Glyphs 90
The Paste Special Command 91
Copying Glyphs to Another Font 92
Appending Glyphs to the Font 93
Copying Composite Glyphs 94
Duplicating Unicode Indexes 95

Deleting Glyphs 96

Creating New Glyphs 97

Highlighting Glyphs 98
Marking Glyphs 98
Glyph Collection 100

Searching for Glyphs 103

Renaming Glyphs 105

Reencoding the Font 107

Unicode-Related Operations 110
Generating Unicode Indexes 110
Generating Names 112
Removing Unicode Information 112

The Font Map Panel 113
Managing Double-Byte Codepages 115

Notes 116

Sorting Glyphs 117

Working with Multiple Fonts 118
Window Menu 119
Fonts Panel 120
Merging Fonts 121
Saving and Opening a Workspace 122

Applying Actions 123

Blending Fonts 124

Exporting Fonts 127
Font Formats 128
Exporting for PC 129
Exporting for Macintosh 131

Export Options 139
Type 1 Export Options 141
TrueType Export Options 145
OpenType Export Options 150
Recommended Export Options 152

Printing Fonts 153

Font Window Options 155
View Options 156
Tuning Options 157

THE FONT HEADER 159

Font Info Dialog Box 160
Command Bar 162
Copying Font Info 163

Font Names 164
Basic Identification and Names 165
OpenType-Specific Names 168
Non-English and Special Names 169
Copyright Information 173
Font Embedding 174
Designer Information 175
License Information 176

Font Identification 177
Version Information 177
Basic Font Identification 178
PANOSE™ Identification 180
Other Identification Systems 182

Metrics and Dimensions 184
Font UPM Value 184
Basic Font Dimensions 185
Advanced Vertical Metrics 187
Superscript and Subscript 190

Encoding and Unicode 191
Supported Codepages 192
Supported Unicode Ranges 194

Hinting Settings 195
Alignment Zones 196
Type 1 Standard Stems 198
Global Hinting Parameters 200

Format-Specific Options 202
Type 1 Export Options 203
TrueType Export Options 204
TrueType Mapping Settings 205
Device-Dependent Metrics 206

Font Smoothing Control 207
[head] Table Settings 209
Basic PCLT options 210
PCLT Identification 211
PCLT Metrics and Font Description 212
PCLT Codepages 213

THE GLYPH WINDOW 215

Glyph Window Contents 216
The Glyphs Bar 219

Selecting a Glyph for Editing 221

Changing the View in the Glyph Window 222
Quick Zoom Selection 223

Tools and Operations 225

Edit Mode 227

Editing Layers 229
Editing Layers Panel 230

Outline Layer 232
Contours 234
Outline Appearance 240
FontAudit 245
Moving Nodes 248
Using the Keyboard 251
Non-node editing 252
Changing Connection Type 254
Deleting Nodes 255
Deleting Lines and Curves 256
Eraser Tool 257
Inserting Nodes 258
Using the Drawing Tool 260
Adding Points to a Contour 261
Converting Primitives 262
Breaking and Joining Outlines 263
Node Commands 264
Node Properties 267
Previewing Glyphs 269

Selections 276
Using the Magic Wand Tool 277
Moving the Selection 278
Copying the Selection 279
Transform Selection 282
Selection Commands 286
Selection Properties Panel 288
Transformation Panel 289
Find and Replace Operation 290

Building an Outline From Blocks 292
Outline Actions 297
Creating Contours 299
Merging and Intersecting Contours 303
Outline Optimization 304

Metrics 305
Editing Metrics 306
Baseline Properties Panel 308
Metrics Properties Panel 308

Grid Layer 309

Guidelines Layer 310
Editing Guidelines 311
Guidelines Popup Menu 313
Guidelines Tracking 314
Guidelines Properties Panel 315

Mask Layer 316
Editing Mask 317
Mask Operations 318
Assigning a Mask 319
Specifying Mask Appearance 320

Template Layer 322

Background Layer 323
Background Positioning 326

Hints And Links Layer 327
Links 328
Editing Hints 329
Hints Tracking 330
Editing Links 331
Hint and Link Popup Menu 332
Hint Commands 333
Autohinting Options 334
Hint Properties Panel 336
Link Properties Panel 336

Anchors Layer 337
Moving Anchors 338
Removing Anchors 338
Renaming Anchors 338
Changing Anchor Color 338
Anchor Properties 339
Using Anchors to Build Composites 340
Using the Anchors Panel 343
Creating Composites and Ligatures 345
Aliases Table 347

Vertical Metrics 348

Alignment Zones 349

Sketch Mode 350
- Moving Points 353
- Changing Point Type 353
- Removing Points 353
- Inserting Points 354
- Reversing Contours 354
- Selecting Points 355
- Moving the Selection 356
- Transforming the Selection 356
- Selection Operations 357
- Breaking and Joining the Sketch Outline 357
- Converting to Outline 358

VectorPaint Mode 359
- Freehand Select Tool 361
- Pen (Contour) Tool 362
- Brush Tool 363
- VectorPaint Options 365
- Line Tool 366
- Polygon Tool 367
- Ellipse and Rectangle Tools 368
- Text Tool 369

Meter Mode 370
- Setting Guidelines, Anchors and Sidebearings 372

Outline Operations 373

Interpolation 374

Envelope 378

Curve 381

Set 382

Reversing a Contour's Direction 383

Rearranging Contours 384

Working with Composite Glyphs 385
- Adding a Component 386
- Decomposing 387
- Component Positioning 388
- Component Properties 390

Using the Primitives Panel 391

Importing and Exporting Glyphs 394

Printing a Glyph 396

EDITING FONT METRICS 397

What are Font Metrics? 398
- Kerning 399
- Metrics Files 400

Metrics Window 401
- Selecting a String for Previewing or Metrics Editing 404

Editing an Underline 409

Editing Metrics 410
- Manual Metrics Editing 411
- Using the Keyboard 412
- Using the Metrics Table 413
- The Measurement Line 415
- Automatic Metrics Generation 416

Quick Save and Quick Open 418

Editing Kerning 419
- Manual Kerning Editing 420
- Using the Keyboard 421
- Using the Table 422
- Using the Kerning Dialog 423
- Automatic Kerning Generation 425
- Resetting Kerning 427

Adjusting Metrics and Kerning 428

Class-Based Kerning 430
- Classes Panel 431
- Class Definition 433
- Defining the Class 434
- Key Glyph 435
- Editing Class-Based Kerning 436
- Kerning Assistance 437
- Metrics Assistance 440
- Class-Based Kerning and OpenType Fonts 442

Opening Metrics Files 443

Saving Metrics Files 445

Printing a Sample String 446

TRANSFORMATIONS 447

The Transform Dialog Box 448

Transformation Actions 451
- Outline Transformation 452
- Hints and Guidelines Transformation 460
- Metrics Transformation 462

Effects 465

Transform Range Dialog Box 471
Transformation Range 472
Transformation Program 474
Using the Preview Window 475
Saving and Opening a Transformation Program 476

Transforming Fonts 477

HINTING 481

Font Scaling, PPM 482

Coordinate Rounding, Gridfitting 483

TrueType And Type 1 Hints 484

Type 1 Hints 485

Font-Level Type 1 Hints 486
Alignment Zones 487
Standard Stem Widths 491
Additional Control Data 494

Stem Hint Programming 496

Type 1 Hinting Tool 499
Inserting And Removing Replacement Points 501
Adding and removing hints 502
Editing Hints 503
Autoreplacing 504
Preview Panel 504

Some Examples 509

TrueType Instructions 510

Font Parameters 512
Visual TrueType Hints 513

TrueType Hinting Tool 514
TrueType Toolbar 516
Layers 518

Options Panel 519

Preview Panel 521

Program Panel 523

Alignment Instructions 526
Alignment Zones 526
AlignTop and AlignBottom Instructions 527
Hinting Alignment Zones 528

Align Instruction 529

Links 532
Standard Stems 533
Single Links 534
Double Links 539

Interpolation 543

Delta Instructions 545
Middle Delta Instructions 546
Final Delta Instructions 551

Removing Instructions 552

Standard Stems 553

Advanced Options 558

Context Menu 559

Hinting Sidebearings 565

Hinting Composite Glyphs 566

Automatic TrueType Hinting 567

Working With Bitmaps 568
Importing Bitmaps 569
Editing Bitmaps 570
Highlight Differences 572
Exporting Bitmaps 573

Hinting Strategies 574
Middle Delta or Final Delta 574
Single Link or Double Link 576
Hinting White Space 578
Hinting Serifs 580
Hinting Diagonals 581
Hinting Symmetrical Characters 582
Interpolate or Not 583

MULTIPLE MASTER FONTS 585

Multiple Master Fonts Theory 586
Design Axes and Dynamic Range 588
Standard Axes 590
Design Coordinates and Weight Vectors 593
Extrapolation 595
Anisotropic Interpolation 596
The Axis Graph 597

Multiple Master Fonts in FontLab 598
Creation of MM Fonts in FontLab 599

Defining an Axis 600
Selecting a Master Font 602
Using an Axis Panel 604
Previewing the Intermediate Design 607
Designing Master Fonts 608
Multiple Master Metrics 615
Editing Axis Settings 616
Removing an Axis 617
Multiple Master and Font Info 618
Editing the Axis Graph 620
Generating a Single-Master Font 623
Expanding the Master 624
Hinting Multiple Master Fonts 625
Generating a Multiple Master Type 1 Font 626

OPENTYPE FONTS 627

Font Features 628
Features and Lookups 630

Scripts and Languages 631

OpenType Font Formats 632
OpenType Tables 634

Feature Definition Language 635
Language Syntax 636

OpenType and FontLab 647

Importing OpenType Fonts 648

OpenType Panels 649
OpenType Panel 649
Adding and Removing Features 651
Reordering Features 652
Entering the Glyph and Class Names 653
Renaming Glyphs and Classes 654
Compiling the Feature Definitions 655
The Output Panel 657
OpenType Features Preview Panel 658
Converting the Kerning 660
Feature Development Process 661

Substitution Lookups 662
Single Substitution 663
Ligature Substitution 665
Alternate Substitution 667
Context-Dependent Substitutions 668

Positioning Lookups 672
Glyph Geometry 673
Value Record 674

Single Positioning 675
Pair Positioning 676

Known Features 679

OpenType Glyph Properties 683
Caret Positioning 684

Exporting OpenType Fonts 685
FontLab and VOLT 687

MACRO PROGRAMMING 689

The Python Programming Language 690

Installing Python 691

Macro Toolbar 692
Assign to Keyboard 694

Integrating into Menus 695

Macro Tool 696

Edit Macro Panel 697
Naming the Programs 698
First Steps 699

FontLab Python Classes 701
FontLab 702
Font 704
Glyph 706

INDEX 709

Introduction

Thank you for purchasing FontLab 4.5 – a new revolutionary version of our professional font editor.

The key features of FontLab 4.5 are:

- Outline editors with more than 20 tools and 200-level undo/redo
- Import and export of fonts in OpenType, TrueType and PostScript Type 1 formats with up to 5000 characters
- Import, export and native-mode editing of Multiple Master fonts
- Import and export of individual glyphs in EPS format
- Multiple Master-enabled metric and kerning editing module with customizable autospacing and autokerning features
- OpenType feature editing and testing
- Import of OpenType features from existing fonts
- Import and export of font metrics files in PFM and AFM format
- Professional-level manual and automatic Type 1 and TrueType hinting
- Automatic transformation of glyphs with more than 25 filters
- Library of predefined customizable graphics primitives
- Automatic testing of glyph outlines with our unique FontAudit technology
- Integrated Macro language based on the industry-standard Python language
- New Unique Sketch mode with easy drawing tools
- VectorPaint tools
- Support of 4 encoding modes and an unlimited number of encoding tables
- Easy-to-use completely customizable drag/drop-based user interface
- Popup menus and property panels everywhere
- Sample printing of fonts, sample strings and individual glyphs
- Automatic Multiple Master-compatible font blending
- OpenType-based Kerning and Metrics assistants
- Smooth outline preview

About this Manual

This manual covers the Macintosh version of FontLab 4.5.

The following chapters describe all of FontLab's features in full detail. They are organised to cover all the functions in their usual sequence.

FontLab User Interface

This chapter covers the basic definitions of the FontLab user interface and its customization and gives a short description of all the FontLab editing windows and panels.

Editing Fonts

This chapter explains how to modify fonts, copy characters, change encoding tables, select characters for editing, and edit font info fields.

The Font Header

This chapter provides a detailed description of the Font Header data and the FontLab tools intended to manage it.

The Glyph Window

FontLab includes powerful outline-editing tools that are described in this chapter.

Editing Font Metrics

If you want to create a professional-looking font you have to edit the font's metric data. The glyphs' widths, sidebearings, and kerning can be edited in FontLab automatically or manually. This chapter shows you how.

Transformations

From scale to drop shadow, from autohinting to autospacing – more than 25 transformation filters can help you instantly expand your font collection. This chapter gives detailed descriptions of all the transformations and their usage in FontLab.

Hinting

To make your Type 1 or TrueType fonts look great everywhere you have to set hints. FontLab includes hinting tools that were previously available only in high-end font editing systems. Hinting can be a complicated process, so read this chapter carefully to get the best results.

Multiple Master Fonts

Opening, editing and exporting Multiple Master fonts; adding and removing design axes; editing the Design Map Graph – everything you ever wanted to know about multiple master fonts is in this chapter.

OpenType Fonts

This chapter covers FontLab tools, panels and features that deal with creation and editing of OpenType font features: ligatures, small caps, fractions, alternative glyphs, etc.

Macro Programming

This chapter includes a short description and demonstration of the Python programming language and its integration into the FontLab user interface. Python can be used to create custom tools and operations within FontLab. A brief description of the FontLab classes exported to Python is provided.

System Requirements

The Macintosh version of FontLab requires the following hardware and software configurations:

- a Power PC based computer capable of running Mac OS 9.1;
- one of the following versions of Mac OS:

  Mac OS 9.1 or later classic operating system (v 9.2.2 is recommended) or

  Mac OS X 10.1 or later (v 10.2 is recommended).

At least 14Mb of free space on the HDD and at least 64 Mb RAM. FontLab will start on 32Mb RAM but you will need more RAM to open bigger fonts.

FontLab User Interface

Before we start talking about fonts and the FontLab font-editing features let's spend some time learning the FontLab user interface. For the most part it is a standard Macintosh interface so if you know how to navigate in Mac OS you will feel comfortable with FontLab. In other parts it is unique and that is where we will focus.

Some of the interface elements in FontLab 4 are completely customizable and from this chapter you will learn how to change the FontLab interface so it will best fit your needs.

Please note that further in the book we will refer to menu commands, toolbar buttons and keyboard shortcuts as they appear in the default FontLab environment under Mac OS X v 10.2, prior to any modifications you may make. All screenshots for the book illustration were also made under Mac OS X v 10.2.

1

Basic Terms

We cannot go any further without defining a few terms that are critical to understanding FontLab and fonts in general.

Character

The minimal unit of the written language – a part of the alphabet, a symbol.

Any picture that can be recognized as having the same meaning represents the same character:

All the pictures above mean the character 'A'

Please note that sometimes pictures that look the same represent different characters:

A Latin 'A'

A Cyrillic 'A'

A Greek 'Alpha'

Characters have *codes* that are used to store text data on a computer.

Glyph

The basic element of the font, literally – an image that is printed. All glyphs are unique, even if they represent the same character.

Glyphs are used to represent characters. Please note that many different glyphs may be used to represent the same character, even in the same font:

Font

An organized collection of glyphs and font header information. Usually glyphs that are united in a font have some similarities in design and other properties.

In the past, a “font” was defined as a single size of the characters of a particular typeface. Now, since fonts are scalable, the term “font” covers all possible sizes of the same typeface design.

Encoding

When text is printed an important process takes place: character to glyph mapping. The source text (in computer form) is a list of codes that represents a list of characters. A font (see above) is a collection of glyphs. So there must be some way to relate characters to glyphs so that when the computer’s operating system encounters a certain character it knows which glyph to print. This “mapping” (or “vector”) is called the encoding. Sometimes the encoding information resides within the font itself as part of the header and other times it is in a separate file.

Font Family

It is important to know the difference between a font and a font family. A font family is a set of fonts that represents some design idea. “Times” is a font family (sometimes called typeface). “Times Bold Italic” is a font.

A font family may include from one to a few dozen fonts.

Glyph name

The only identification of a glyph (other than its visual appearance) is its name. A Western glyph name consists of Latin characters, digits and punctuation. It is highly recommended you name glyphs in accordance with the following rules:

1. No spaces.
2. No digits at the beginning.
3. Only '.' And '_' punctuation marks are allowed in the name.

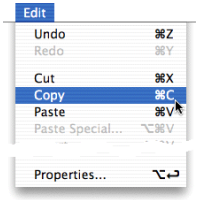
Menu

When we refer to menu items in the main FontLab menu, we will use the following notation:

[top menu item]/[sub-item]

For example:

Edit/Copy means: click the word Edit on the menu bar and select the **Copy** command from the menu:



Directories

When we have to refer to one of the folder created during FontLab installation, we will use the following syntax:

```
FontLab/[directory name]
```

Where FontLab is the name of the folder where you have installed FontLab 4 (usually /Applications/FontLab4) and [directory name] is the name of the folder, as in the example below:

```
FontLab/Data
```

Mouse

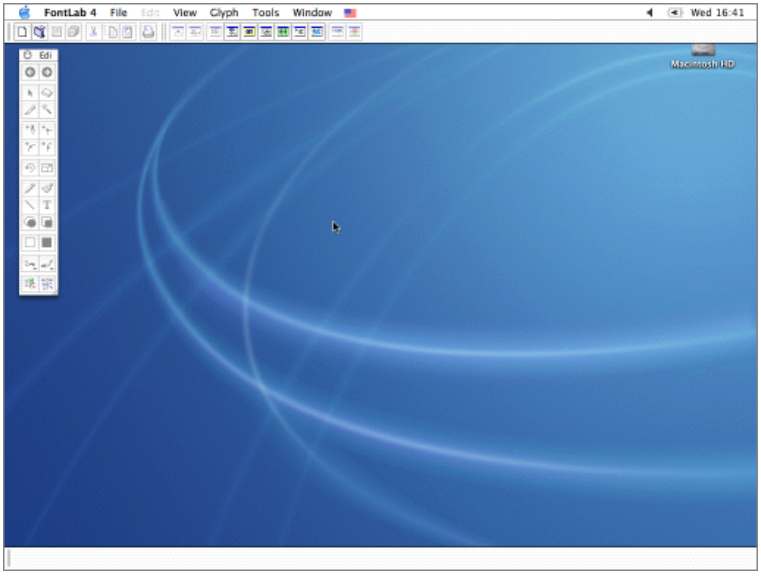
Click the mouse on some object	Position the mouse cursor on the object and click the mouse button
Ctrl-click something	Position the cursor over "something", hold down the **CTRL** key on the keyboard and click the mouse button. If you have a mouse with two buttons, you may use the right mouse button instead of pressing the **CTRL** key.
Drag some object	Position the cursor on the object, press the mouse button and move the mouse to move the object. Release the mouse button when you're done.

Context Menu

Most windows and panels in FontLab have attached context menus. To open the context menu, **CTRL**-click (or click the right mouse button if present) an empty area in the window or panel. Some context menus may have the Macro submenu at their bottom if the Python system is installed.

Getting Started

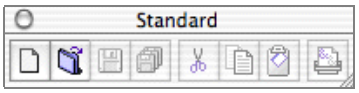
When you run FontLab 4 for the first time (to run FontLab double-click on its icon) you will see a welcome screen for a few seconds and then the FontLab window:



Like almost all Macintosh programs FontLab has a menu, a few toolbars and a status bar at the bottom.

There is nothing special about the FontLab menu except that there is a difference between the Classic and Mac OS X environments. In Mac OS X there is an **Application** menu containing the **Quit** and **Preferences** commands. Under Mac OS 9 the **Quit** command resides in the **File** menu while the **Preferences** command resides in the **Tools** menu.

The usual location of toolbars is at the top of the screen, but if you want to put them somewhere else, just drag them there:



The status bar can be placed only at the bottom (default) or at the top of the screen. Some tool specific toolbars are floating only and cannot be docked to the sides of the screen.

You can easily choose which toolbars you want to see: use the **Toolbars** submenu in the **View** menu.

Following is a list of all common toolbars with a few comments about each:

Standard	Contains basic commands like file open and save, copy/paste and print
Panels	Controls the appearance of FontLab *panels* – shared windows used to control most professional FontLab features
Tools	Gives access to editing modes that you will use to work on glyph shapes and operations from the Tools/Operations menu
Edit	Probably the most important – a mixed toolbar containing the main editing tools and frequently used operations
Macro	Gives quick access to pre-written *macro programs* that can automate various font-editing tasks. It is available if the Python 2.2.1 system is properly installed
Show Layers	Controls the appearance of basic *Editing layer*. It is analogous to the View/Show Layers menu
Lock Layers	Allows one to lock/unlock the Editing layers. It is analogous to the View/Lock Layers menu
Outline	Contains commands from the Tools/Outline menu
Hints & Guides	Contains commands from the Tools/Hints & Guides menu
Templates	Contains commands from the Tools/Templates menu
Operations	Contains commands from the Tools/Operations menu
Multiple Master	Contains commands from the Tools/Multiple Master menu
Mark	Contains commands from the Mark submenu of the Font window's context menu.

You may notice a few *italic* terms. We will describe them later. Specifically, *panels* and *Glyph window* will be described in a few pages; *Editing layers* in the "Glyph Window" chapter; and macro programs in "Macro Programming".

OK, we are almost ready to open a sample font, but before we do let's talk about *customization* of the FontLab user interface.

Customizing FontLab's User Interface

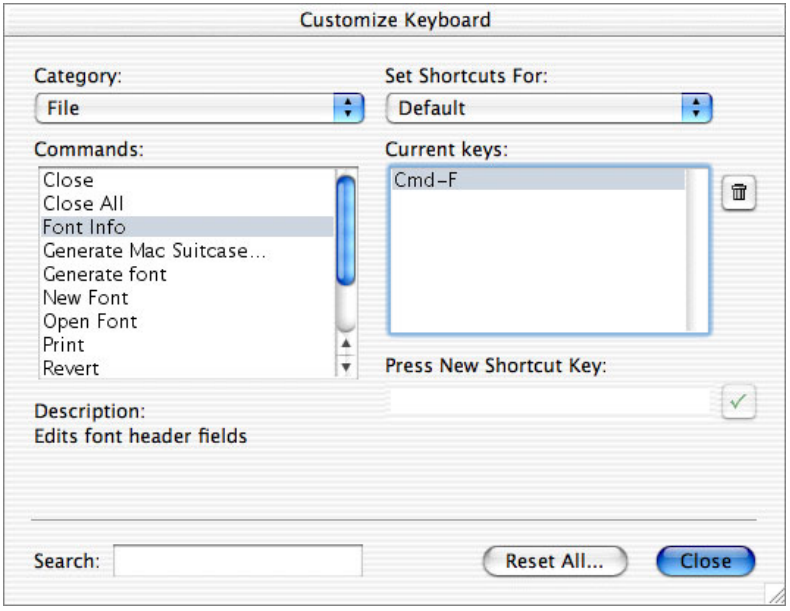
As you may infer from the title of this section some of the FontLab user interface (which means the toolbars and keyboard shortcuts) is customizable. We believe our default interface is the easiest to use, but if for some reason you don't like it, you are free to make any changes you want. If you don't want to change anything in the FontLab user interface, you can fast forward to the next section.

You cannot customize menus in FontLab 4.5 for Macintosh yet. The customization of toolbars is also restricted to switching them on or off or arranging them on the screen.

The general idea of keyboard customization is simple: there is a long list of commands that you can use with keyboard shortcuts. Through customization you can assign any command to a combination of keys pressed on the keyboard.

Customization of the Keyboard

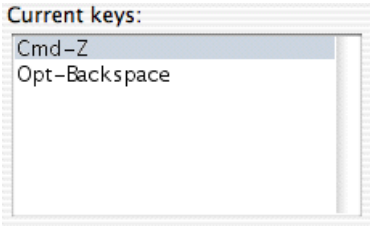
Select the **Tools/Customize** command and the **Customize Keyboard** dialog box will appear:



In the left area of the page you can select the command that you want to customize. Choose the commands category in the top popup menu and the command itself in the list below. You will see a short command description below the commands list.

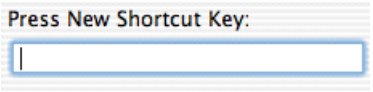
To easily find a particular command by its name, type it in the **Search** field at the bottom of the dialog box.

On the right part of the page you will see the list of keyboard shortcuts currently defined for that command:



The button at the right of the list allows removal of one of the existing shortcuts.

To define a new keyboard shortcut, select a command and position the cursor on the editing field below the **Press New Shortcut Key** label:



When the editing field has focus just press the combination of keys that you want to assign. A description of that combination will appear in the editing field and you can click the **Assign** button **to assign that combination to the currently selected command**.

Press the **Reset All** button **to reset all changes back to FontLab defaults**.

Press the **Close** button at the bottom after you finish customizing the keyboard shortcuts to save changes and close the dialog box.

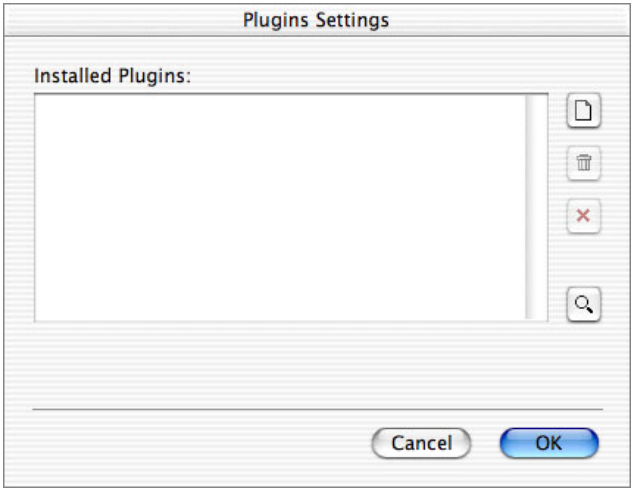
Links to External Programs

Some applications available from FontLab Ltd. have a common communication interface which allows them to easily exchange font related data. We call programs that can accept data from other programs *FontLab Server Applications*. Similarly, programs that can send data to FontLab servers will be referred to as *FontLab Client Applications* or plugins. And programs that combine server and client capabilities are called *FontLab Client/Server Applications*.

FontLab itself can work as server and sometimes as client as well. For example, it can accept bitmap and outline data from BitFonter (our bitmap font editor) or send outlines to BitFonter for rasterization.

To make FontLab communicate with plugins installed locally and to have instant access to them from the FontLab menu, you have show FontLab where they are.

Use the **Tools/External Tools/Settings** command to find other applications that can work together with a FontLab Server. The Plugins Settings dialog box appears:



There is a list of the installed plugins in the dialog and it is empty by default. Click this button: **to add a link to the program that you know to be a plugin**. Select the plugin application in the standard Open File dialog box and press **Open**. The application name will appear in the list.

An easier way is to let FontLab find the plugins itself. Press the button at the right of the list and FontLab will automatically search for plugins on the available disks.

The button at the right of the list allows removal of the link to the plugin selected in the list.

Press the **Reset All** button **to clear the list of plugins**.

After you press the **OK** button the plugins added to the list will appear in the **Tools/External Tools** menu to let you quickly launch them when needed.

FontLab Windows

There are only three types of Windows in FontLab:

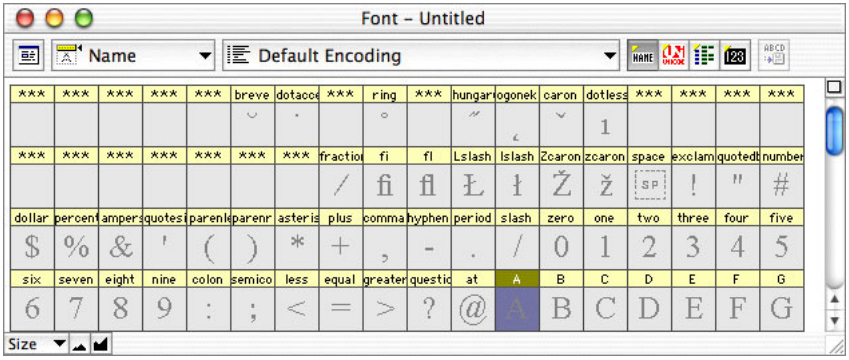
Font window	Represents one of the opened fonts
Glyph window	Used to edit glyphs
Metrics window	Used to edit glyph metrics and kerning.

In this chapter we will provide only very basic information about the main windows. Please refer to the “Editing Fonts”, “Glyph Window” and “Editing Font Metrics” chapters to get detailed information about the windows and their features.

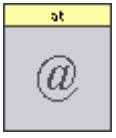
Font Window

As an exercise let's create a font in order to demonstrate the FontLab Windows. Use the **New** command in the **File** menu or click this button on the Standard toolbar:

You will see the Font window:



As you can see, this window has a caption with a few buttons and options and a big table of cells that represent characters and glyphs. Each cell has a caption that contains glyph identification information: name, Unicode index or some other data:



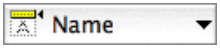
Cells can also contain little icons that show properties of glyphs, but more about that later.

There are no glyphs in the font that we just created, but the Font window nevertheless shows some pictures in the glyph cells. These are template images that show which character should be placed in the cell. FontLab has templates for thousands of characters, so you will usually know where to place new characters.

We'll discuss navigation in the Font window later, in the Editing Fonts chapter, so let's talk about the Font window command bar, which is located at the top of the window:

On the command bar you can easily find a button on the left which is a duplicate of the **File/Font Info** command, which is described later in this manual.

Right of the buttons there is a popup menu, which allows you to change the information that appears in the character cell's captions:



Next is a popup menu that allows you to change the encoding table of the current font:

We will talk about encodings later, but you could choose a couple different ones from the popup menu and see how the Font window changes.

At the right of the encoding list there are four buttons that allow a choice of encoding modes. Again, a detailed description of this follows. Just a few words here: any glyph in the font may be identified by a name, Unicode index or just its order in the glyph table. Four buttons in the command bar allow you to choose one of four modes: Names, Unicode Ranges, Codepages or Index.

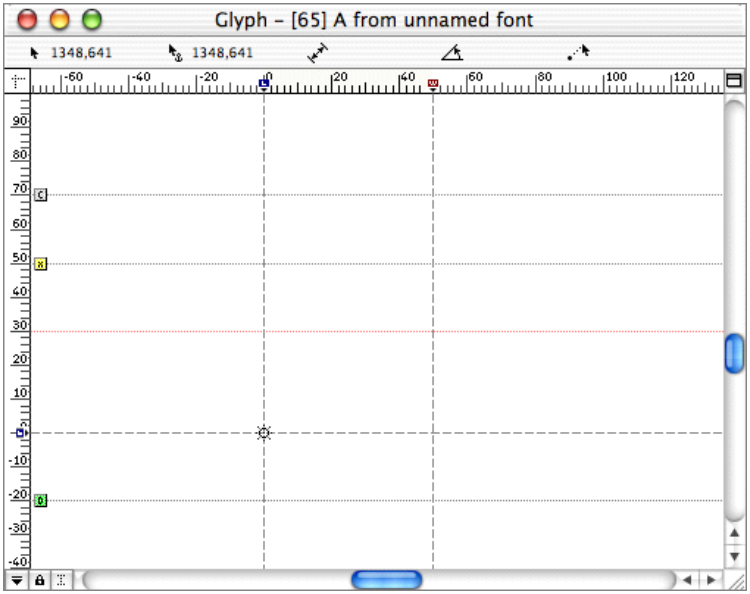
The last button controls the saving of custom encoding file.

That's all about the Font window for now so let's open the Glyph window.

Glyph Window

To open a Glyph window for editing individual glyphs you need to create one. Remember, we started with a new font that doesn't have any glyphs. **To create a glyph**, double-click on any cell in the Font window. You will see that the gray cell (which means there is no glyph defined) is replaced by a white one, which represents a glyph that is defined, but contains no image. When you draw or paste something into it, the white cell will show a small picture of the glyph.

After the glyph cell is created we are ready to open the Glyph window. Select the glyph cell (just click on it with the mouse button) and double-click it to open the Glyph window. It will immediately appear on screen:



Instead of double-clicking, you can also use several other methods **to open the Glyph window**:

1. **Ctrl**-click the glyph cell and select the **Edit in new window** command in the context menu.
2. Select the glyph and choose **New Edit window** in the **Window** menu.
3. Select the glyph and press on the Panels toolbar.
4. And finally, select the glyph cell and just press the **Enter** key on the keyboard.

If you have more than one glyph in your font (which is normal when you open an existing font) and have a glyph window already open when you double-click another glyph in the Font window (or use any other method of opening a glyph window except the **New Edit window** command or a button on the toolbar). A new glyph will appear in the original glyph window. If you need to open many glyph windows simultaneously just hold down the **Cmd** key when you double-click the new glyph cell or otherwise open a new Glyph window.

You may have as many open glyph windows as you want, just close those you don't need so as not have all your workspace covered with glyph windows. You may use the **Cmd-W** shortcut to close windows. If the current window is the Font window closing it will close the font.

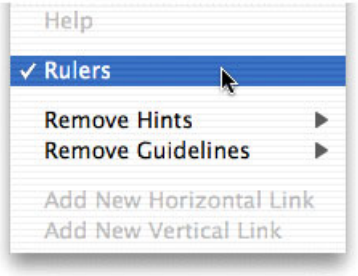
Glyph Window Contents

All windows in FontLab have a similar layout: control panel on the top and main area covering most of the window. The glyph window is no exception. Click on the small button on the top of the vertical scroll bar: ▭ and the Glyph window control area will appear. The control area contains zoom selection tools, a popup menu, and a few toolbar buttons:

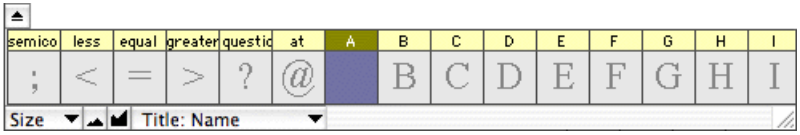
Actually there are two toolbars here: the left one is used only to select the zoom mode of the Glyph window and to choose the **Zoom in** and **Zoom out** commands and **Undo** or **Redo** commands. The second toolbar is used to select the properties of the editing tool and is hidden by default. We will describe it later.

The main area of the window has scroll bars to change the view of the glyph, and vertical and horizontal ruler bars.

You can switch the ruler bars on and off with the **Rulers** option in the **View** menu. A quicker way is to **Ctrl**-click the ruler and choose the option in the context menu:



At the very bottom-left corner of the Glyph window you will find a little expand button that, when clicked on, opens the *Glyphs bar*:

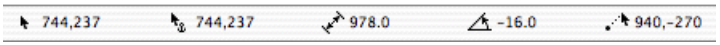


This is nothing more than a slice of the Font window, making it easier to access cells in the font window while the glyph window is maximized for precise editing work.

At the bottom-left corner of the Glyph windows you will find two more buttons, Lock and Meter:

The Lock button controls quick access to the font glyph – when it is in the "unlocked" state you can use the keyboard to directly access the glyphs. I.e. when you press a key the corresponding glyph will automatically open in the glyph window.

The Meter button controls the appearance of the Meter bar, which sits above the glyph window ruler and shows the current coordinates and other parameters of the cursor:



We will return to a more detailed description of the glyph window properties in the "Glyph Window" chapter.

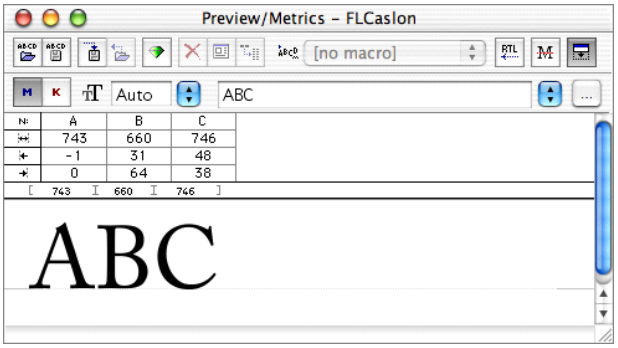
Finally, let's quickly preview the last window in FontLab: the Metrics window.

Metrics Window

The Metrics window is used to adjust glyph metrics – glyph sidebearings and kerning. Open any existing font from the disk, e.g. FREESERF.VFB from the FontLab/Samples folder.

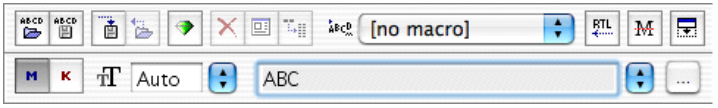
To open the metrics window select some glyphs in the Font Window and click on the **New Metrics window** command in the **Window** menu (**CMD-OPTION-SHIFT-M** shortcut).

You will see a new window:



Glyphs that are currently selected in the Font window or the glyph that is in the active glyph window will appear in the metrics window.

The metrics window has a main editing field, scroll bars, and command area with a local toolbar:



To choose a string of characters to preview or modify use the string selection control:

The popup button to the right of the control allows you to choose a pre-defined string. To the right of the button there is an options button. Click it to get access to the Metrics windows **Options** dialog box where you can customize the window.

One powerful option in this dialog box is support for a second preview string. The second string appears below the main preview string and can be used to compare different characters. The second string is not directly editable.

Metrics Window Toolbar

The Buttons on the toolbar mean:

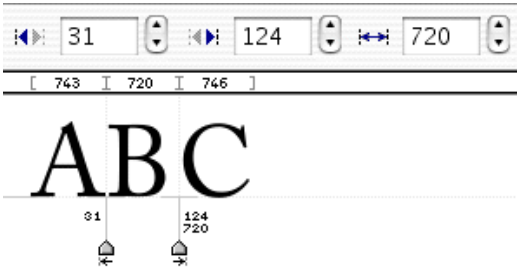
	Open metrics file	With the metrics window you can import metrics and kerning information from AFM, PFM or MMM (in the case of a Multiple Master font).
	Save metrics file	You can save AFM, PFM, AMFM and MMM files to use them later with other fonts.
	Quick save	Click this button to store current information about metrics to a special temporary file. If you do it before any serious modification of the metrics you can easily undo changes.
	Quick open	Restores metrics data saved with the Quick save command.
	Auto	Automatically calculates metrics or kerning.
	Reset kerning	In Kerning mode may partially or completely remove information about pair kerning.
	Kerning dialog	In Kerning mode opens the dialog box where you can preview and manage kerning pairs.
	Kerning Assistance	Opens the Kerning Assistance dialog box.
	Macro preview	Allows selection of special macro programs that can be used to simulate complex environments to test metrics and kerning.
	Right-to-left	Turns the Metrics window into right-to-left writing mode (for work on Arabic or Hebrew scripts).
	Measurement line	When this button is pressed, all metrics are measured with respect to the measurement line.
	Expand	Controls the compact or expanded appearance of the editing field.

Metrics Modes

The metrics window works in three modes: *metrics, kerning* and *preview*.

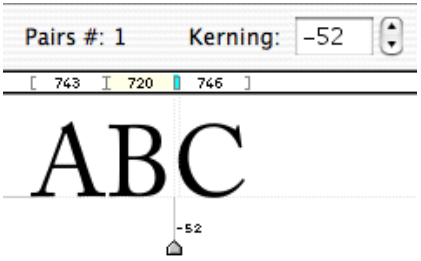
By default the Metrics window appears in expanded Metrics mode. To contract the Metrics window, use the collapse button .

In **Metrics mode** you can change the glyph sidebearings using either visual or digital controls:

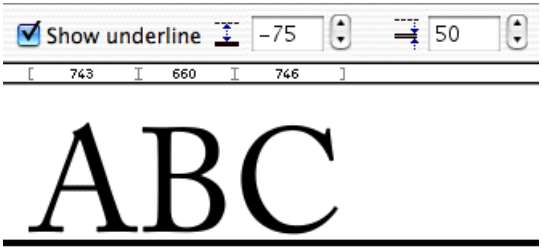


In Metrics mode the string of glyphs is previewed without kerning.

In **Kerning mode** you can change pair kerning:



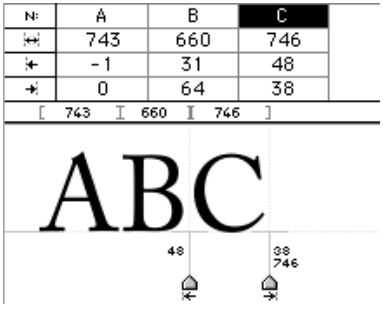
In **preview mode** (M and K buttons depressed) you cannot change glyph metrics or kerning, but you can safely preview the font in high-resolution mode. In addition, you can preview and modify the position and width of the underline line:



The editing field of the Metrics window may be in compact or expanded mode. By default it is expanded.

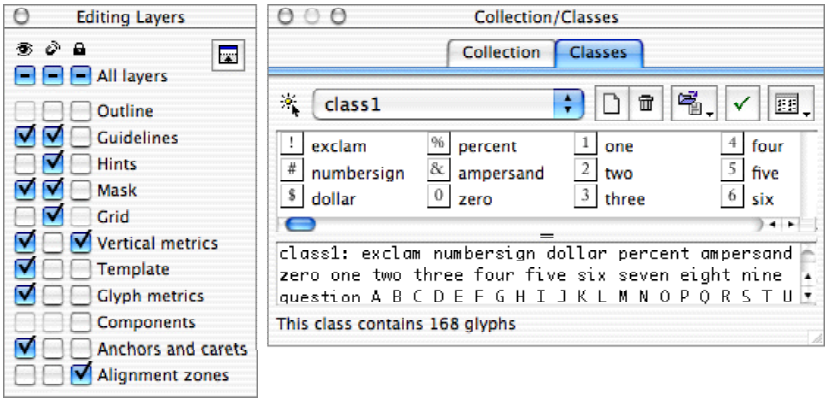
To expand the metrics window, use the expand button .

When the window is in expanded mode, the bottom line of the control bar disappears and the metrics bar is expanded and contains information about the metrics of the glyphs that are currently visible:

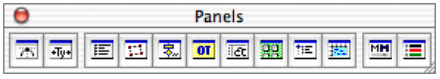


Panels

Some FontLab operations are accessible through Panels – small windows that are located in front of the main Font, Glyph and Metrics windows:



Use the **Window** menu or **Panels** toolbar to open panels:



Below is the list of all the panels available in FontLab. They are described in full detail in the sections that are related to their functions, so this is only a short reference:

Editing Layers	Control of all editing layers, “show”, “snap”, “lock” operations
Transformation	Panel for digital outline transformations
Edit Macro	Editor for Python macro programs
OpenType	Editor for OpenType features
Output	Text output panel. Other panels and macro programs may output text here. This panel is not available in the Panels toolbar.
Preview	Preview, OpenType Sample and Anchor preview panels.
Collection/Classes	Classes and Collections: named or unnamed lists of characters
Fonts	List of all opened fonts grouped by family name
Font Map	A picture representation of big Unicode fonts
Axis	Selector of intermediate (or extrapolated) design in a Multiple Master font
Masters	Selector of master in a Multiple Master font
Primitives	Collection of outline primitives. This panel is not available in the Panels toolbar.

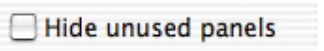
All panels are described in full detail in the following chapters when we discuss the features that they serve.

All the panels can stick to either side of the screen and to each other. **To make a panel stick** just drag it close to the screen or another panel’s edge.

To prevent the panel from sticking, hold down the **Cmd** key while dragging the panel’s title bar or switch off the option in the **Preferences** dialog box:

With this button you can quickly enlarge a panel. Click the button again **to return the panel to its original size**.

Some panels are available only under sertain conditions. You may choose to hide those that are out of use or leave them always open. Use the checkbox in the **Preferences** dialog box to control the panels' behaviour:



Every time you exit FontLab the positions of all toolbars and panels are stored in preferences file, so when you run FontLab the next time, the environment will be restored.

Editing Fonts

In this chapter we will discuss the editing of fonts. A font is a collection of glyphs with similar design and some encoding and header information. The information includes font identification names, copyright data, character encoding information and other data that is necessary for font usage.

2

Opening Fonts

With FontLab you can create new fonts or open existing fonts for modification. When you open an existing font, however, please be sure that modifying it does not violate copyright laws: some fonts are copyrighted as software so it is not legal to change them in any font editor. Carefully read the license agreement that comes with every font.

You can find two (one serif and one sans-serif) royalty-free, non-copyrighted fonts that you can use as a basis for your own fonts or characters in the FontLab/Samples folder on your CD or on our website (www.fontlab.com).

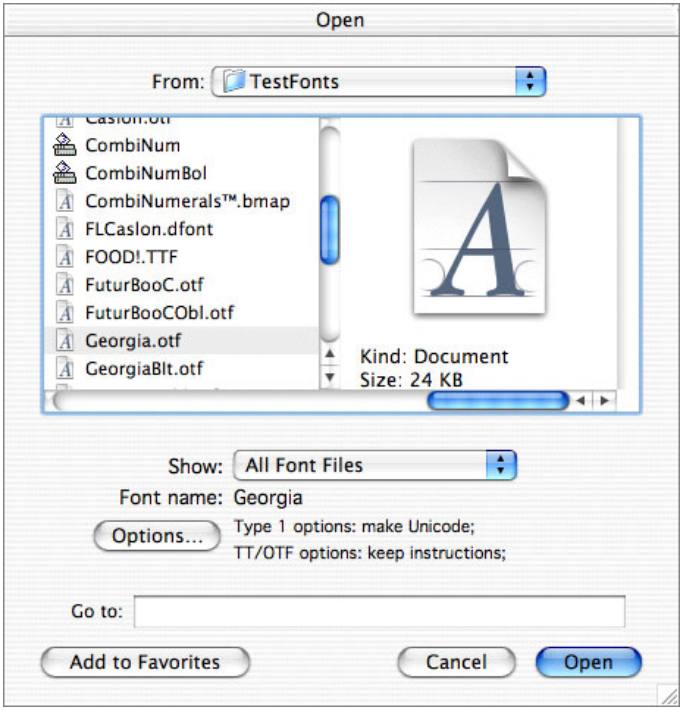
Please, note that with FontLab you cannot export fonts that contain more than 5000 characters. You can open big fonts, but only a portion of them can be saved or exported. If you need to work with really big fonts, we recommend you try our AsiaFont Studio product:

http://www.fontlab.com/html/asiafontstudio.html

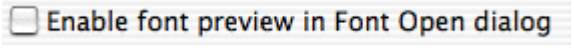
To open a font for editing, select the **File/Open** command, or

click the button on the Standard toolbar.

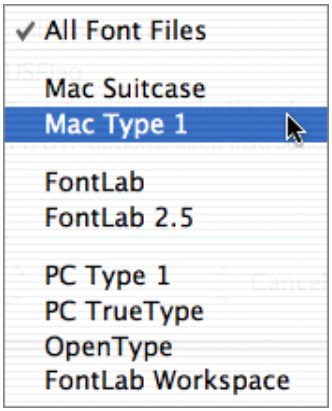
You will see the standard **Open File** dialog box in which you can select a font file to open. Note that in this dialog box you will see all the fonts that can be imported: Mac Suitcase (without extension or .dfont), Mac Type 1, PC TrueType (TTF), PC Type 1 in binary (PFB) and ASCII form (PFA), OpenType (OTF) and fonts in FontLab formats (VFA and VFB).



To see the preview of the font in the Open dialog box, use the **Enable font preview in Font Open dialog** option located on the General page of the **Preferences** dialog box:



If you want to list only fonts in a particular format, select that format in the popup menu located in the center of the **Open** dialog box:



When you select a font file in the files area, you will see the font name in the bottom of the dialog box.

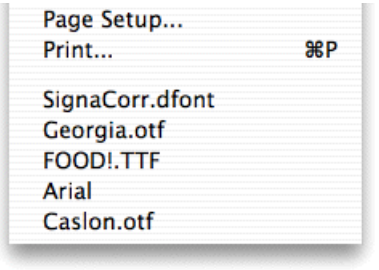
You can open multiple fonts with a single operation: just select them in the list with **Shift**-click or **Cmd**-click.

The current import options appear below the font name field. You can set the options here by clicking the **Options** button. We will discuss the import options in full in one of the next sections.

To open fonts installed in Mac OS you are better off copying the font files into some other directory and then opening them in FontLab from that directory. Or you can simply drag-drop the font files from the Finder window into FontLab.

Most Recently Used Fonts

All fonts that you recently opened in FontLab are added to the bottom of the **File** menu:



Next time you want them, just select the font file in the **File** menu and FontLab will open it.

Opening Fonts with Drag-Drop

The easiest way to open fonts in FontLab is to drag-drop font files from Finder. Even if FontLab is not running, you can drag-drop files onto its icon on the desktop or in the Dock to run FontLab with those fonts open.

Font Formats

FontLab can import outline fonts in any of following formats:

Format	Extension	Description
Mac Suitcase	none or dfont	TrueType fonts in Macintosh-compatible resource-based form or Mac OS X-compatible data fork-based form
Mac Type 1	none	Single Master or Multiple Master PostScript fonts in Macintosh-compatible resource-based form
PC Type 1 Binary	PFB	PostScript fonts in Windows-compatible binary form
PC Type 1 ASCII	PFA	PostScript fonts printer downloadable form (this format is also used in most Unix-based operating systems)
PC Multiple Master	PFB or PFA	Special version of PostScript fonts — in binary or ASCII form
OpenType	OTF	CFF-based OpenType fonts.
PC TrueType	TTF	TrueType and TrueType-based OpenType fonts that are standard in Windows and Macintosh operating systems
FontLab 2.5	VFA	Fonts in FontLab 2.5 internal format
FontLab 3.0	VFB	Native format of FontLab 3.0, 4.0 and 4.5

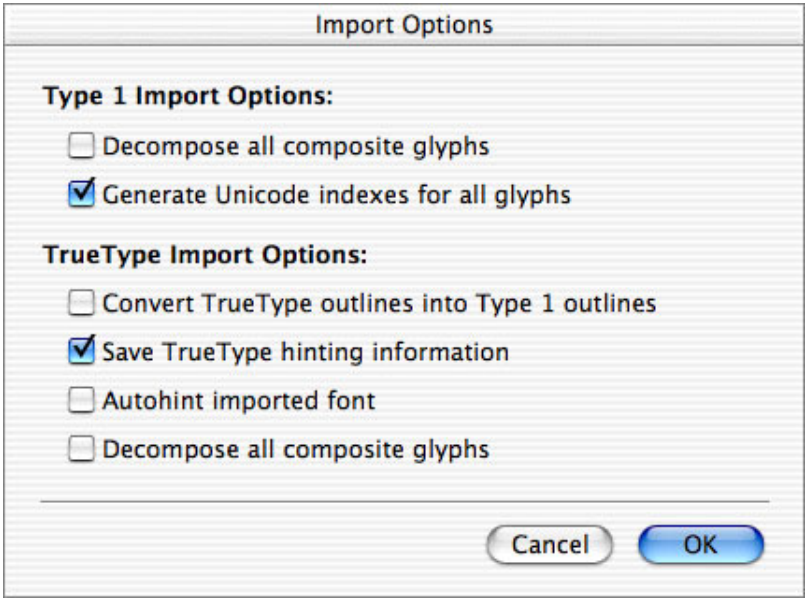
Note that FontLab 4.5 cannot open font files that were created in FontLab 2.0. The default extension of those files is VFA (same as FontLab 2.5 fonts), but they are not compatible with FontLab 4.5. If you want to move fonts from the FontLab 2.0 format to FontLab 4.5, use PC Type 1 Binary as an intermediate format.

If you have fonts in a different font editor's own font format and you want to open these fonts in FontLab 4.5, export the fonts in the Type 1 Binary format using the other font editor and then open the Type 1 fonts in FontLab.

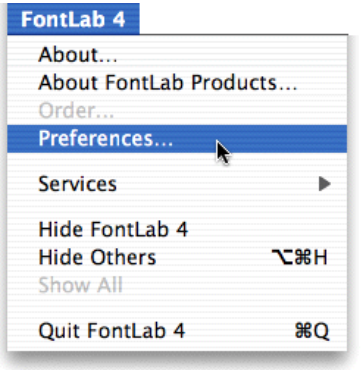
Import Options

Before opening font files you may want to set some options that can help you get started more quickly. In the following sections we will discuss all the font-opening settings. We will use some technical terms that will be discussed in more detail later, but if you want to check the definitions now you can find them in the Index.

All the opening options are accessible through the **Import Options** dialog box. **To open this dialog box** use the **Options** button in the middle-left area of the **File Open** dialog box:

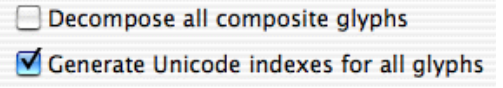


The same options are available in both **Type 1** and **TrueType** pages of the **FontLab Preferences dialog box** accessible through the **Preferences** command of the **Application** menu:



Type 1 Import Options

There are two options in opening Type 1 fonts: **Decompose all composite glyphs** and **Generate Unicode indexes for all glyphs**.



If the first option is on then FontLab will decompose all composite glyphs in the imported font. Composite glyphs have no unique outline themselves, but "borrow" outlines from other font glyphs. Good examples of composite glyphs are accented glyphs, like 'À', 'å' or 'ñ'. In each of these the composite character is composed of a character glyph outline and an accent glyph outline from elsewhere in the font. FontLab has all the necessary tools and operations to work with composite glyphs, so it's usually not necessary to decompose them on import. But if you want to markedly rearrange the glyphs and do not want to worry about composites you can use this option. Of course you can always decompose or recompose the glyphs later using FontLab commands.

When FontLab opens TrueType fonts it always decomposes composite glyphs that have rotated or slanted components.

The second option is usually on. We strongly recommend keeping it that way if you plan to convert your Type 1 font to the TrueType format. The TrueType font format uses Unicode indexes to access characters, so having the indexes set properly is paramount. However, if you do not plan to make a TrueType font you may switch this option off. As in the case of the first option, you can always make Unicode indexes later.

How FontLab Makes Unicode Indexes

In the FontLab/Mapping directory you will find the file STANDARD.NAM. This file contains a database that links the PostScript names of characters with Unicode indexes. When you import a Type 1 font and the option Generate Unicode indexes for all characters is on FontLab takes the name of every imported character and looks for it in the names database. If it locates the name there it takes the associated Unicode index and adds it to the character's list of indexes.

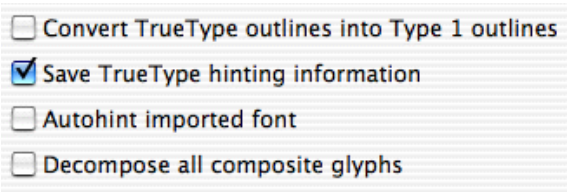
Note 1: The Names' database has more than 4000 records and includes almost all known names for all European, Cyrillic, Arabic and Hebrew languages and for most symbol and dingbats fonts.

Note 2: The names' database is a text file that can be edited. You can add new records to this file at any time. Be very careful when you edit this file because incorrect records may make exported fonts unusable in some environments.

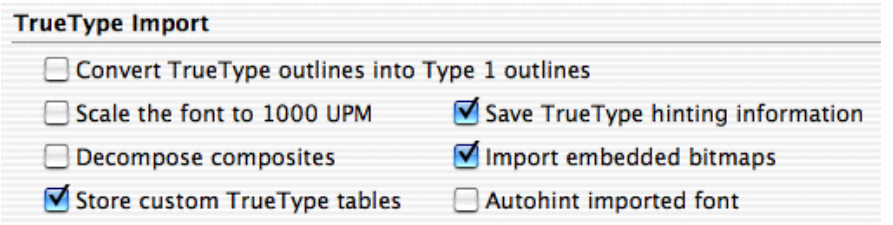
Note 3: It is possible to link more than one Unicode index to a name and more than one name to a Unicode index. If FontLab finds several indexes linked to the name, it will assign all the indexes to the character. (Refer to the Encoding Modes section for a description of the multi-Unicode indexing method.) If more than one name is linked with the Unicode index then FontLab will take the first one it finds as the name for the known index.

TrueType Import Options

The **Import Options** dialog box that appears if you click the **Options** button has the basic set of TrueType import options:


Convert TrueType outlines into Type 1 outlines
Save TrueType hinting information
Autohint imported font
Decompose all composite glyphs

The full set is available on the **TrueType** page of **the Application/Preferences** dialog box:



Convert TrueType outlines to Type 1 outlines

In FontLab you can work with Type 1 (based on 3rd -order Bezier curves) or TrueType (based on 2nd-order splines) outlines. If you open a TrueType font and plan to export this font as TrueType, it's better to keep the original outlines to avoid conversion errors. But if you plan to export your font as a Type 1 font, then we recommend converting the outlines on import. You can convert the outlines in either direction at any time using FontLab commands.

Scale the font to 1000 UPM

Typically TrueType fonts have UPM (Units Per eM – the size of the grid on which all glyph coordinates are defined) equal to 2048. Type 1 and Type 1-based OpenType fonts have UPM equal to 1000. You can change the UPM value at any time using the FontLab commands, but if you turn this option on, UPM will be converted during the font import.

Save TrueType hinting information

Leave this option on if you want to keep the original TrueType instructions and outlines. FontLab will keep the imported TrueType data until you change the glyph's outline or hints. If you import a TrueType font to rearrange glyphs or to add some new glyphs we highly recommend saving the original TrueType hinting data.

Autohint imported font

To prepare an imported TrueType font for Type 1 editing and export, you may ask FontLab to automatically make Type 1 hints for all the glyphs. FontLab will use the current Type 1 hinting settings and will make hints for TrueType or Type 1 outlines depending on the conversion setting (**Convert TrueType outlines to Type 1 outlines**)

Decompose all composite glyphs

This option works exactly the same as for Type 1 fonts.

Import embedded bitmaps

If this option is on, FontLab will read all embedded bitmaps defined for the source TrueType font. You can edit them using the TrueType hinting tool and optionally export to TrueType format. Embedded bitmaps may help to improve font readability at low point sizes and in some cases can be used instead of TrueType hinting.

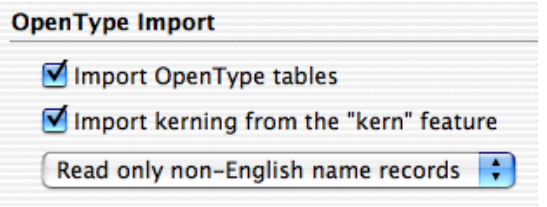
Store custom TrueType tables

Some TrueType fonts have additional tables that are not a part of the TrueType or OpenType specification. If you want to import these tables and have them exported unchanged when the font is saved, switch on this option.

This feature is very useful if you are working with additional tools, like Microsoft VOLT or VTT programs. If the "Store custom TrueType tables" option is active, FontLab will not change or destroy tables that these tools include in TrueType fonts.

OpenType Import Options

Special options for importing OpenType fonts can be customized on the **OpenType** page of the **Application/Preferences** dialog box:



Import OpenType tables	If you want FontLab to read OpenType tables (GSUB, GPOS and GDEF), keep this option on.
Import kerning from the "kern" feature	In OpenType fonts kerning data is stored as feature named "kern". This option lets you import kerning data from that feature and to convert it to plain pair kerning format. You can perform this operation later using FontLab's OpenType-editing tools, but we recommend doing it during font import.
Name-reading options	To support multiple languages OpenType and TrueType fonts often have several versions of the font names. Use the choices of this control to select one of the options to read those names.

Creating a New Font

If you want to create a new font from scratch, you select the **New** command from the **File** menu. FontLab will create an empty font that will not have any characters and will open an empty Font Window.

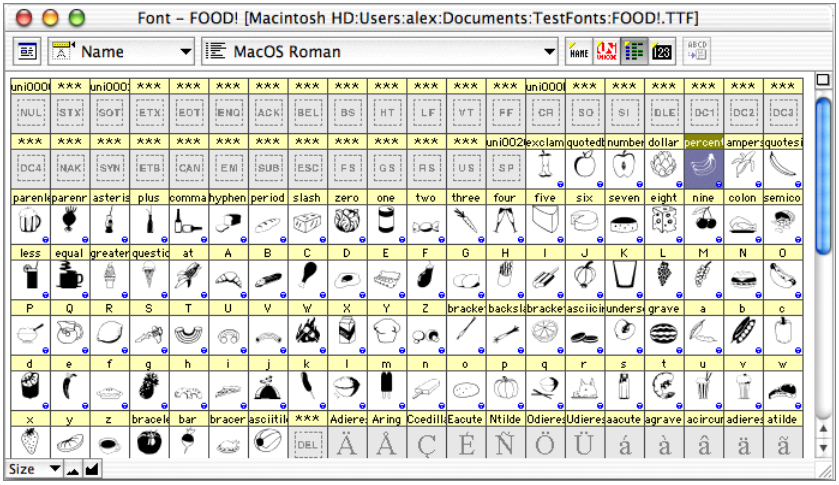
You may then add new glyphs. To make your font usable you must also fill in the important Font info fields. (See the “Font Header” chapter.)

Font Window

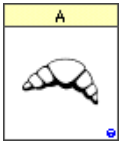
The Font Window is used to display an entire font. It opens automatically when you open an existing font for editing or choose to create a new font.

In FontLab you can open many fonts at once and every font will have its own Font Window. The Font Window is a representation of the font, so when you close this window the font will also close.

You can do a lot of things using the Font Window — from browsing a font for a desired character to rearranging and remapping the font to editing the Font info fields. The following sections of this chapter will tell you how to use this window.

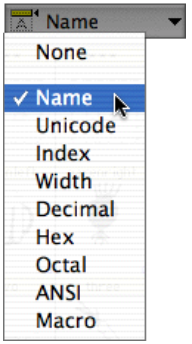


The Font Window consists of the *header*, which includes a few buttons and two popup menus, a *footer* and a *table of characters* (*character chart*), where each character is represented by a single cell:



Each cell has a *caption* showing the name of the character or its code in various forms.

The **left popup menu** located on the Font Window header area:

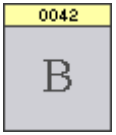


lets you **select one of the caption modes**:

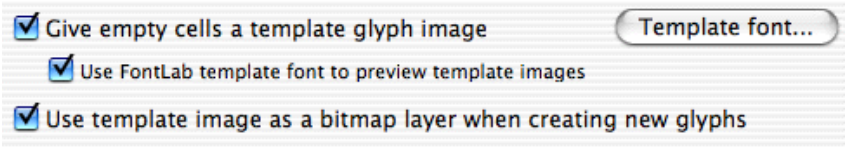
None	The caption is not shown
Name	The Character name will appear in the cell caption
Unicode	The Character's Unicode index
Index	The Glyph's index
Width	The Character's width
Decimal	The Decimal character code
Hex	The Character code as hexadecimal number
Octal	The Character code as octal number
ANSI	The ANSI character that corresponds to the character code
Macro	The appearance of the caption depends on the special macro program. By default this option shows the number of components in composite characters.

The characters' cells may have a gray or white background and a gray or yellow caption.

A gray cell means that there is no character defined for this place in the font. I.e. the character does not exist in this font. Instead of the character, a sample template character from one of the system fonts or from the FontLab template font is shown in the empty cell.



You can select a font for the templates or switch off the templates using the **Font Window** page in the **FontLab Preferences** dialog box:


Give empty cells a template glyph image
Template font...
Use FontLab template font to preview template images
Use template image as a bitmap layer when creating new glyphs

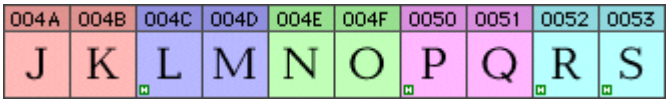
A yellow frame and caption in a character's cell means that this character is in the currently selected encoding, codepage or Unicode range (see next section). Characters that are not in the current encoding have a gray caption and no yellow frame.

The small marks that appear in the corners of the character cell mean:

Left-Top	Blue mark	Character has more than one Unicode index assigned
Right-Top	Yellow-red or yellow-green mark	Character has compatible mask layer
Left-bottom	Green or red 'H' mark	Character has hint replacement program or overlapping Type 1 hints. A Green mark means that the program is correct.
Right-bottom	Brown or blue 'T' mark	Character has TrueType hints, either original (blue mark) or manually set visual (brown mark)

The meaning of all the marks will be discussed in full detail later.

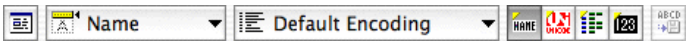
Some characters may be *marked* with a different color for the caption and background:



Marking is very useful when you need to show visible differences among characters for easy identification. More about that later.

Font Window Header

At the header of the Font Window you see one button on the left and five buttons in the right area:



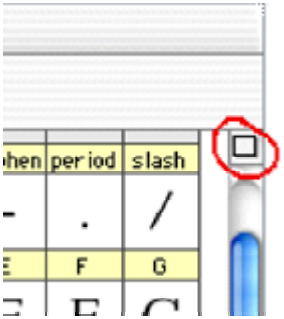
The left button opens the **Font Info** dialog box for the current font. This is the same as choosing the **Font Info** command in the **File** menu.

The first four buttons on the right allow you to select one of the encoding modes:

	Switch to the Names mode
	Switch to the Unicode Ranges mode
	Switch to the Codepages mode
	Switch to the Index mode

The last button in the Font window header area is used **to save current encoding**.

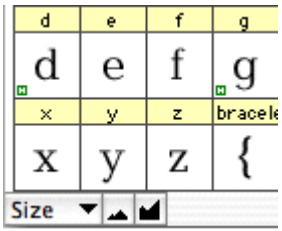
The Font window header can be switched off. To do this just click on the small button at the top of the vertical scroll bar:



You will see the header collapses, leaving more space for characters.

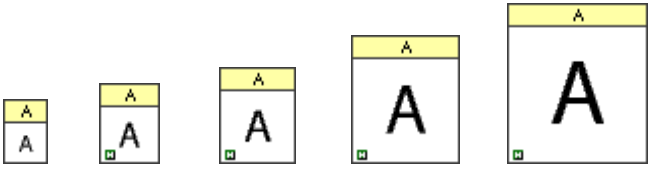
Font Window Footer

At the bottom of the Font Window you see a footer with one popup menu and two buttons at the left end:



Use these controls to customize the font window view. Select the desired size of glyph cell in the **Size** popup menu. Possible sizes vary from 16x16 up to 64x64 pixels. Smaller cells occupy less space but hide details. If you select the smallest size (16x16) you will not be able to see the additional marks which are visible in cells at larger sizes.

A sample of the different cell sizes:



16x16 24x24 32x32 48x48 64x64

Use the **Zoom In** and **Zoom Out** buttons to enlarge or reduce the cells' size.

When the font window's header is switched off, the footer shows three additional popup menus:

The **Title** popup menu is used to change the cells' caption contents. The **Mode** popup menu corresponds to 4 buttons in the window header. And the third popup menu is the synonym for the Encoding menu in the window header.

Encoding Modes

Support for almost all known character indexing methods is one of the key FontLab features.

Here's how it works:

A font is just a big collection of glyphs that are used to represent many characters (more about that in the following section). On an average screen the Font Window can show just a few hundred character cells, so we need to have some method to browse the font "through" the Font Window.

In FontLab you can choose one of four so-called Encoding modes that allow you to select a subset of the glyph collection and show it in the top part of the Font Window for easier access.

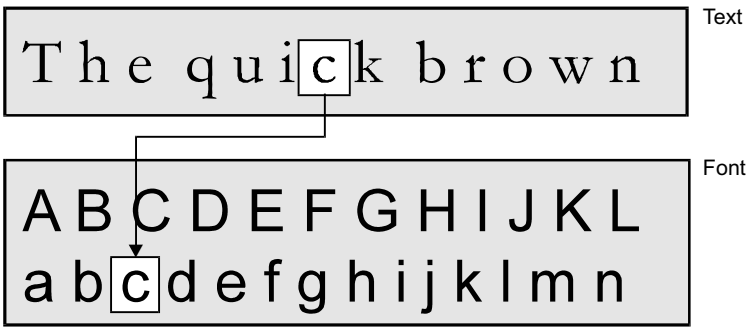
In the following sections you will find more information about encoding modes, Unicode and name-based identification and the character-glyph model.

Characters, Codes and Glyphs

A font is a collection of glyphs that usually have a common design. Each font has a header that stores top-level information, such as the font name and style, in an organised way.

Character Mapping

Text in digital form is a collection of *codes* — integer numbers. When text is entered into a computer we (with the help of a text processor program) replace characters with codes. When text is printed or previewed on the screen, the printing program performs the reverse process — replacing codes with character shapes (glyphs) that it gets from the font.



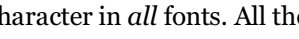
A character *mapping standard* is a set of rules that defines the relation between characters and the codes that are used to represent these characters in the computer. There are many mapping standards used in the world to help use different languages. The main difference between mapping standards is the size of the code. There are one-byte, double-byte and multi-byte mapping standards. With a one-byte mapping standard only 8 bits of information is used to represent each character, so it's possible to use only 256 characters.

A Double-byte mapping standard uses two bytes (16 bits) for every character. So it's possible to map 65536 characters. Multibyte mapping standards use from one to four bytes for every character — expanding the code space to billions of characters.

One-byte encoding standards are enough to represent most Western languages, but they run out of room if more than one language needs to be supported, or in the case of Eastern languages that use very big alphabets. Even double-byte encoding standards cannot cover all the characters of the world. The most well known and widely used double-byte encoding standard is *Unicode*.

Character and Glyph Identification

To simplify access to the characters in the font each character has an *identifier* — an integer number or a text name. All font formats allow you to quickly locate a character whose identifier is known. A printing or previewing program that wants to output text converts character codes to character identifiers in the font. This process is usually performed using a special index table, called the encoding table, which defines the relationship between character codes and identifiers.

Note that the term *character* covers *all* possible appearances of the character in *all* fonts. All these "pictures":  are the same character 'A' — the first letter of the English alphabet. "Instances" of characters that have a unique design are called *glyphs*. But note that the same picture 'A' may mean different characters – the first letter of the English alphabet, the first letter of the Russian alphabet and the first letter of the Greek alphabet can be the same glyph, but are not the same character. Precise character identification methods are really necessary.

There is a difference between character and glyph identification. When we need to identify a character it means that we are identifying all glyphs that may be used to represent that character. If we are identifying a glyph, it is unique.

Characters are identified by integer number. To support exchange of information this number must be assigned according to some standard. In FontLab all characters are identified according to the Unicode standard.

A glyph is identified by its name or by its index – its sequential number in the collection of glyphs in the font. Since the index may vary (for instance, if we remove the glyph with index 5 the indexes of all glyphs that have larger numbers, starting from 6, will change) it is not very convenient, so in most cases glyphs are identified by names.

It is very important to understand that there is no one-to-one relation between characters and glyphs. It is possible to have several characters (and character identifiers) refer to a single named glyph. On the other hand, some glyphs in the font may not be characters and may not have a character identifier.

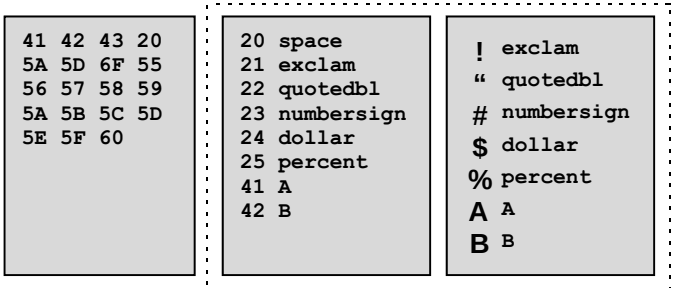
Character Identification in FontLab

In FontLab there are four modes of character and glyph identification:

1. **Names mode.** This mode historically is used in Type 1 fonts. An interesting point is that Names mode may have two different applications: it may be used to reference **glyphs** or **characters**. In the latter case all referenced characters must have a one-to-one relationship between character and glyph, so there is no difference between referencing the character and the glyph. Tables that are used to reference glyphs or characters in Names mode are called Encoding Tables. There are two kinds of these tables: a mapping encoding table references characters in Type 1 fonts and is based on one of the Adobe standards; while a definition encoding table references glyphs and has no relation to characters. This is sometimes useful in managing a font at the design stage.
2. **Codepages mode.** This is a **character**-identification method – a special table of Unicode indexes is used to map a subset of characters to the top part of the Font Window table. This table, called a codepage, may use one or two bytes to represent a character mapping record. Two-byte codepages are used to reference characters in Far-East fonts: Chinese, Japanese, Korean or Traditional Vietnamese.
3. **Unicode Ranges mode.** Unicode is a two-byte standard so it potentially can cover 65,536 characters (not taking into account the extension methods implemented in the Unicode 3.1 standard). This is a huge code space and to manage it most efficiently it is divided into several ranges. Usually a range is designed to cover a single script, like Cyrillic, Armenian or Thai. The Unicode Ranges mode references **characters**.
4. **Index mode.** This is the simplest glyph identification mode: all glyphs are sorted in their sequential numerical order in the collection of glyphs.

Mapping Encoding Tables

The Encoding Table defines links between character codes and names, assuming that all characters in the font have name-based identifiers. The table is loaded and stays stored in memory while the font is used. Character selection processing consists of two stages — conversion of the code to a character identifier and searching the font for the location of the character description with the known identifier:



The Encoding Table can also add more flexibility to fonts. You can put many more characters into one font, assign a unique name to each character, and supply several encoding tables, allowing you to select different sets of characters in the font when you use different encodings.

For example, in symbol fonts the Greek characters take places that are usually occupied by Latin characters. With the encoding tables you can include both sets of characters. Just assign the correct names (like alpha for the ‘A’ character and A for the ‘A’ character) and later you can choose the symbol encoding to work with the Greek version of your font or choose Roman encoding to use the Latin characters.

In FontLab you can include up to 5,000 characters in a font. You can also select any of the predefined encodings to examine a font and then include it as the default encoding upon export.

You even can create your own custom encodings and use them to properly assign names to your characters.

Names Mode

To switch the Font window to the Names mode, click the button on the Font window toolbar or select **Names** in the **Mode** popup menu in the Font window footer.

To select the current encoding for the font, use the **Encoding** popup menu. When you open it you will see the following available encodings:

Imported	The encoding that was loaded with an imported font.
Default	The default encoding for all the applications in the FontLab family. This provides the best support for Windows or Mac OS, depending on the version of FontLab.
Adobe Standard	Adobe Standard Encoding. This is useful if you want to create a PostScript font.
ISO Latin 1	Adobe ISO Latin 1 standard encoding.
Macintosh	Standard encoding for Macintosh computers.
Adobe Cyrillic	Standard encoding for supporting Cyrillic characters on PostScript printers and in Windows.
Macintosh Cyrillic	Standard encoding for supporting Cyrillic characters on Macintosh computers.
Symbol	Standard encoding for supporting fonts that include mathematical and other characters.
ANSI	Standard encoding used in Windows.

Additional Encodings

All encodings are stored in text files that can be edited in any text processor. This is not recommended but may help if you find that our standard encodings do not work for you.

If you want **to define a non-standard encoding**, save the file in a FontLab/Encoding directory:

How to Create a Custom Encoding File

To create a custom encoding file:

1. Duplicate one of the .enc files located in the **Encoding** directory to use as the basis for your new encoding file.
2. Open the copy in any text editor (SimpleText or TextEdit will do) and then edit it, following the same structure that you find in the original file.
3. Change the name of the encoding and the encoding index in the first line of the file. The first line should have the following structure:

   %%FONTLAB ENCODING: 7; Adobe Symbol Encoding

   "%%FONTLAB ENCODING: " is the prefix of the file used to detect properly made encoding files and must not be changed. Note the space between ':' and the encoding index.

   '**7**' is the index of the encoding vector. You must not change the encoding vector indexes of any of the encoding vectors or they will become unusable. If you make your own encodings the indexes of your files should not be used in any of the other files. The actual value of the index is not important, so you can assign indexes like 1001 or 10001.

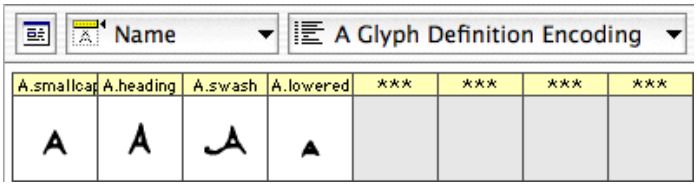
   The last part of the first line, "**Adobe Symbol Encoding**", is the name of the encoding vector. It starts at the first non-space character after ';'.
4. Save this encoding file with a different file name but be sure to use the .enc file extension. Put this file into the directory where all the other .enc files are located.

The new encoding will appear in the **Encoding selection** popup menu the next time FontLab is used.

When you change the encoding you will see that the characters in the Font Window are rearranged. Some characters will move below the "yellow" zone. Remember that only characters that are in yellow cells are covered by the currently selected encoding vector.

Definition Encoding Tables

This kind of encoding table has no reference to the character encoding and is used primarily to organise a collection of glyphs in some useful way. For example, if your font contains several glyphs representing the 'A' character, like "A.smallcaps", "A.heading", "A.swash", "A.lowered", it could be a good idea to have them visually close in the Font Window:



You can do this by creating an .ENC file that enumerates the characters that you want to arrange. Note that it is not required to put any codes into a definition-encoding table, so the body of the .ENC file might look something like this:

```
%%FONTLAB ENCODING: 1001; A Glyph Definition Encoding
A.smallcaps
A.heading
A.swash
A.lowered
```

You can use this form of the encoding table as a template for your font so that you will not miss an important glyph. Please remember, however, that you cannot use this table for any kind of encoding. You must switch to one of the mapping encoding tables or to Unicode-based Ranges or Codepages mode to check the encoding of your font.

The Unicode Standard

The *Unicode Worldwide Character Standard* (that's the full name of the Unicode standard) is a character coding system designed to support the interchange, processing, and display of the written texts of the diverse languages of the modern world. In addition it supports classical and historical texts of many written languages.

In the Unicode standard a character is identified by a double-byte index. The standard potentially can cover 65535 characters in a "basic plane" and much more using plane switching. In the current version of the standard (3.2) several hundred thousand indexes are defined. That covers almost all currently used languages, some historical languages and many pictorial characters.

FontLab can only work with Unicode indexes from the basic plane (which potentially can cover 65,535 indexes).

The Unicode standard is used in TrueType fonts as the main character identification method. In principle TrueType fonts may be encoded with other standards, but in Windows Unicode is always used.

Until version 8.5 the Mac OS had very limited support for Unicode. Only the first 256 characters in the font could be encoded – which made Unicode unusable. Since version 8.5 the Mac OS fully supports Unicode fonts in applications that need this support.

The Macintosh version of FontLab has special TrueType export options that make possible the creation of Mac fonts that will work on both old and new operating systems. We will talk about this later, when we discuss TrueType exporting options, which are slightly different on Windows and Macintosh.

We recommend visiting the Unicode Consortium official Web site at:

http://www.unicode.org

to get more information about this standard.

Unicode Ranges

In Unicode the standard character space (indexes 0 — 65,535) is divided into *ranges*. Each range typically covers characters that belong to one alphabet or have common properties, like the Cyrillic range or the Hebrew or the Extended Latin.

Ranges may be of various lengths — from a few characters to several thousand characters (in the case of Kanji characters).

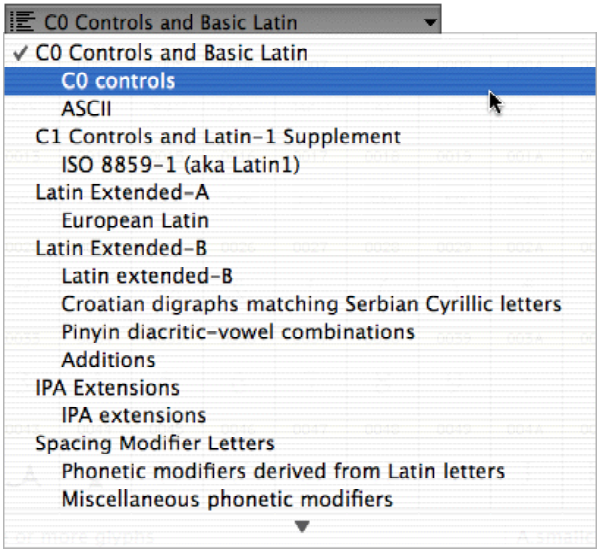
In FontLab you can select any Unicode range and view your font as organized by the range. All characters with Unicode indexes in the selected range will be arranged in order in the yellow zone at the top of the Font window.

In order to simplify working with Unicode ranges in FontLab all the "official" ranges in the Unicode standard are subdivided into subranges. You can work with the whole range or select one of the subranges. For example, you can select the whole Cyrillic range that includes all currently used and historic Cyrillic characters, or you can select just the historic letters or only the Russian alphabet.

To select a range in the Font Window:

1. Switch the Font Window to the Unicode Ranges mode by clicking the **Ranges** button or selecting **Unicode ranges** in the **Mode** popup menu in the Font window footer.

2. The encoding selection popup menu will show the names of all available Unicode ranges and subranges:



   Range names are aligned to the left of the list box and the subranges' names are indented to the right.

3. Select the range or subrange that you want to work with and you will see the Font Window change so that only characters from the selected range are in the yellow (encoded) area.

The definitions of the Unicode ranges and subranges may be changed. They are placed in a text file that you can edit in any text editor. You can add your own ranges or subranges that, for example, may include only one character whose placement is very important to you.

How to Edit the Unicode Ranges Definition File

The Unicode ranges definition file is located in the **Data** directory and has the name uranges.dat. To change this file, open it in a text editor. You will see the following text:

```
%%FONTLAB UNICODE RANGES
0x0000,0x007F,C0 Controls and Basic Latin
0x0000,0x001F,    C0 controls
0x0020,0x007F,    ASCII
0x0080,0x00FF,C1 Controls and Latin-1 Supplement
0x00A0,0x00FF,    ISO 8859-1 (aka Latin1)
0x0100,0x017F,Latin Extended-A
0x0100,0x017F,    European Latin
0x0180,0x024F,Latin Extended-B
```

The first line of this file is an identification line. It should not be changed or FontLab will not accept this file as a valid Unicode range definition file.

All other strings have the same structure:

<first index of the range>, <last index>,<range's name>

Note that there is no space before a range's name but there are four spaces before a subrange's name. Using this simple method you can indent ranges' names as you wish.

Codepages

Codepages are tables that map character codes (one byte long) to the Unicode indexes. Depending on the size of the page, these tables may have 256 or 65,536 records, one for each possible character code. Long codepages are called double-byte codepages and are primarily used to represent codes used in Chinese, Japanese, Korean or Vietnamese languages.

Codepages are necessary because we need to somehow encode text written in different languages in the one-byte code space. So when we have a text file encoded according to some codepage, we use the codepage table to find which characters were used in this text. We may have two different texts with the same code 192 (decimal), but in one case it may mean the Russian 'А' and in the other case it may mean 'À' (Agrave).

Codepages are used not only to identify characters, but also to simplify text sorting, conversion of lowercase to uppercase characters, spell-checking and in many other applications where it is necessary to know which characters are used in the text.

Because the Unicode character identification standard covers most languages it is usually used as the destination information in the codepage tables. Here is an example of fragments from two different codepages that map the same codes to different Unicode indexes:

MacOS Roman	MacOS Cyrillic
0xC0 0x00BF	0xC0 0x0458
0xC1 0x00A1	0xC1 0x0405
0xC2 0x00AC	0xC2 0x00AC
0xC3 0x221A	0xC3 0x221A
0xC4 0x0192	0xC4 0x0192
0xC5 0x2248	0xC5 0x2248
0xC6 0x2206	0xC6 0x2206
0xC7 0x00AB	0xC7 0x00AB
0xC8 0x00BB	0xC8 0x00BB
0xC9 0x2026	0xC9 0x2026
0xCA 0x00A0	0xCA 0x00A0
0xCB 0x00C0	0xCB 0x040B
0xCC 0x00C3	0xCC 0x045B

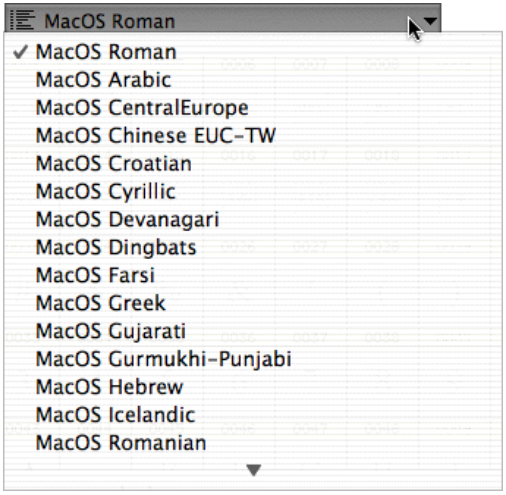
Many different codepages have been defined for many languages and different operating systems. FontLab 4.5 includes descriptions for 124 codepages — all the known Mac OS, Windows, OS/2, MS DOS codepages plus a few others like the Polytonal Greek, Russian KOI-8 and NeXT Step codepages.

In FontLab a codepage is a filter through which you can “look” at your font to see how it will work in different environments. For example, you might include many Unicode characters in your font and see how it would work if it was installed in OS/2 with the Arabic language selected. This gives you the opportunity to easily create fonts that will be properly encoded and will always work correctly.

To select a codepage in the Font Window:

1. Switch the Font Window to the Codepages mode by clicking the Codepages button in the header or by selecting **Codepages** in the **Mode** popup menu in the Font window footer.

2. The encoding selection popup menu will show the names of all available codepages:



   Mac OS codepages come first, MS Windows codepages follow. All other codepages are sorted according to their names.

3. Select the codepage that you want from the list and you will see the Font Window change. All the characters that are in the codepage appear in the "yellow" area. All other characters are in the "gray" area below. Select the MS Windows 1252 Latin 1 codepage and you will see how your font will look in the Windows standard (Western) codepage.

All codepages in FontLab are defined in editable text files, so you can change any codepage if you think it is wrong (please let us know!) or you can define your own codepage. We do not recommend changing any of the codepages supplied with FontLab. They are extensively tested and are based on the documents from the companies who supply them.

Double-byte

If your font contains many characters from one of the Far-East languages you may need to use double-byte codepages. If you select one of these codepages, you will see an additional control to the right of the codepage selection list:

This control allows you to select a “page” of the codepage. Theoretically, we may have 256 pages of 256 codes each, which gives us 65,636 codes. In practice none of the known codepages has that many codes and usually less than half of that number.

How to Make a Codepage Definition File

Codepage definition files (extension CPG) are text files that have the following structure:

%%FONTLAB CODEPAGE: 0xffff; MacOS Cyrillic

%% source: MacOS 8.0b5
0x00 0x0000

0x01 0x0001

0x02 0x0002

0x03 0x0003

The first line of this file is an identification line that is used to set the codepage name and tell FontLab that this file is a properly composed codepage definition file. This line must be started by the text:

%%FONTLAB CODEPAGE: 0xFFFF;

The name of the codepage follows.

All other strings starting with ‘%’ are comments and are not interpreted by FontLab.

The following strings are formed as pairs of two integer numbers in decimal or hex (starting with “0x”) form. The first number is the code of the character and should be in the 0-255 range. The second number is the Unicode index of the character and should be in the 0-65535 (0-FFFFh) range. The special Unicode index 0xFFFF is used to define codes that are not mapped to any character.

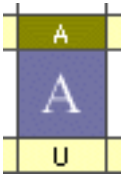
Using the Font Window

The glyph chart in the Font window is a visual representation of all the glyphs in the font. To modify the font you have to learn how to use the glyph chart: navigate, select glyphs and select commands.

From here on we will reference all characters as glyphs. We would like to clarify once more: a font is a collection of *glyphs*. Text is a collection of *codes*. An alphabet is collection of *characters*. Since we are talking of fonts, we will be using the term “glyph”.

Navigating

One of the glyphs in the Font window is the “current” glyph. It is specially highlighted:



To view different parts of the font in the Font window you can either use the vertical scroll bar or the auto-scroll mode: if you place the mouse anywhere in the chart; press the mouse button; and move the mouse cursor above the top or bottom of the chart it will scroll up or down accordingly.

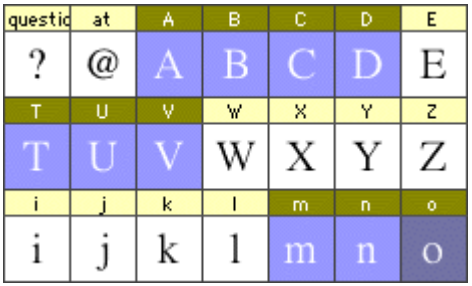
You can also use the spacebar to scroll the Font window. Press the spacebar and drag the mouse to scroll the window vertically.

Alternatively you can use the keyboard keys to navigate in the font chart:

Arrow keys	Moves the current glyph highlight one cell right, left, up or down, according to the key used
Option+Right arrow	Moves 2 cells right
Option+Left arrow	Moves 2 cells left
Page Up and Page Down	Moves the glyph highlight one screen up or down
Home	Moves the glyph highlight to the leftmost glyph on the current row
End	Moves the glyph highlight to the rightmost glyph on the current row
Cmd+Home	Moves the glyph highlight to the first glyph on the chart
Cmd+End	Moves the glyph highlight to the last glyph on the chart

Selecting

In addition to the current glyph you can select sets of glyphs in the font chart. These selections behave similarly to selected text in a text editor – you can copy selected glyphs to another place in the font or to a different font; you can apply different effects to the selection; etc. Selected glyphs have inverted colors:



To select one or more cells, press the mouse button on the first or last cell of your selection and drag the cursor across the cells you want to select. You will see the selection highlighted. If you drag the cursor outside the visible part of the chart, it will scroll accordingly. **To cancel your selection**, click on any glyph cell.

Alternative: Using the navigating keys on the keyboard, set the current cell highlight on the first (or last) cell of a selection, then press the **Shift** key. Move the current cell highlight (as described earlier) to select the cells.

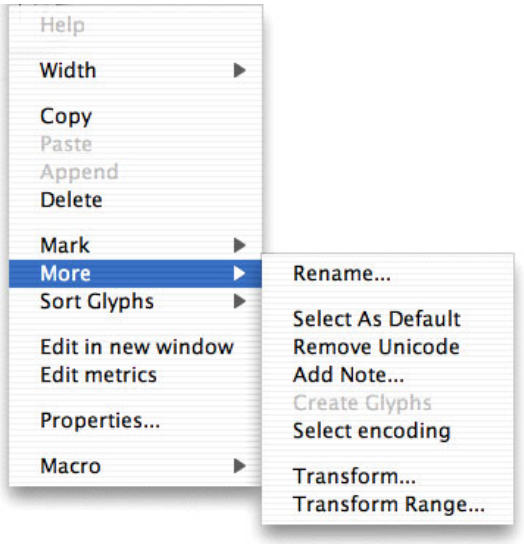
Selection does not have to be continuous. If you press the **Cmd** key, you can select cells in any order and combination.

Context Menu

Most commands available in the Font window can be selected from the popup menu.

To open the popup menu, Ctrl-click mouse anywhere in the chart area.

Here is a sample of the Font window popup menu:



Here is what the commands mean:

Width	Allows you to easily select one of the predefined widths of the Font window.
Copy	Copies the selected glyphs onto the Clipboard. Same as the **Copy** command from the **Edit** menu.
Paste	Places glyphs from the Clipboard into the font starting from the first selected cell. Same as the **Paste** command from the **Edit** menu.
Append	Appends glyphs from the Clipboard to the current font.
Delete	Deletes the selected glyphs. Same as the **Delete** command from the **Edit** menu.
Mark	Marks the selected glyph(s) with one of the predefined colors.
More	Submenu with more commands (described below)
Sort glyphs	Applies different sorting methods to the glyph collection. Note that this has no effect on glyph encoding or names. Only the indexes of glyphs are changed.
Edit in new window	Creates a new Glyph window and opens the current glyph in it.
Edit metrics	Opens the Metrics window with the currently selected glyphs.
Properties	Opens the glyph properties panel for the current glyph or selected glyphs.
Macro	Submenu with macro commands (described later).

Contents of the More submenu:

Rename	Opens a rename dialog box.
Select as default	Selects the current glyph as the “default glyph” that is used in Type 1 fonts to represent glyphs that are not present in the font.
Remove Unicode	Removes the Unicode indexes in the selected glyphs.
Add Note	Adds a Note to the current glyph.
Create Glyphs	Creates a new glyph in every selected empty cell.
Select encoding	Selects the glyphs that are in the current encoding – the yellow area at the top of the font chart.
Transform **Transform Range...**	Opens one of the transformation dialog boxes. Refer to the “Transformations” chapter for more detailed information about available transformations.

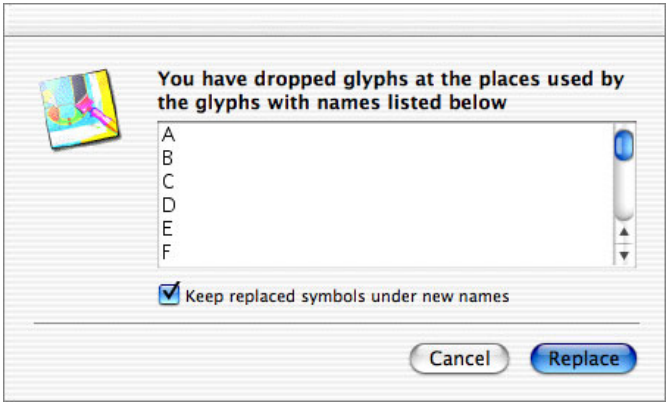
Moving Glyphs

You can change the positions of glyphs in the font chart just by moving them to a new place.

To move glyphs in the font chart:

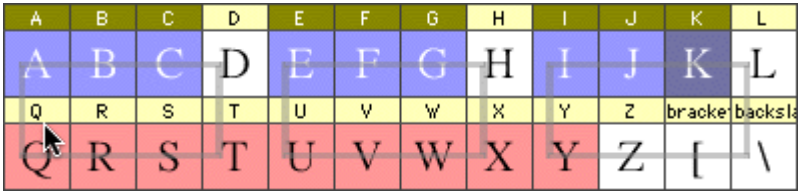
1. Select the glyphs that you want to move.
2. Position the mouse cursor on the selected glyphs.
3. Press the mouse button.
4. Drag the glyphs to the new position. Release the button to finish moving or press the **Esc** key to abort.

If you move glyphs over the cells of existing glyphs, you will see a dialog box prompting you to choose whether to replace the existing glyphs or save them by moving them to the end of the encoding:



Leave **Keep replaced symbols under new names** checked to save the glyphs (I.e. put the new glyphs in the cells and move the existing glyphs to cells at the end of the encoding) or clear it to replace them (I.e. delete the existing glyphs).

Note that even if source selection is not continuous the destination selection *will* be continuous:



If you are working in the Index mode, when you move glyphs they will not replace existing glyphs that they overlay. Instead moved glyphs will be inserted in front of the existing glyphs in the glyph sequence of the font.

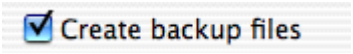
Saving the Font

Most of the font-modification operations are not undoable, so we recommend you save your work regularly.

To save a font that you have opened from an existing font file (in FontLab or other format), use the **File/Save** command or click on the **Save** button on the Standard toolbar.

To save all opened fonts click the **File/Save all** command or this button on the Standard toolbar: .

Font(s) will be saved in FontLab format (vfb extension) to the directory where the original font was opened (imported). If this option in the **General** page of the **Preferences dialog box** is active:



FontLab will create a backup copy of the existing font file before overwriting it with the new one.

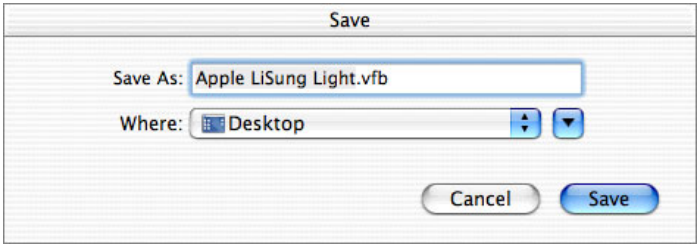
If you are working with a new font or you want to select the destination directory or change the name of the file, use the **File/Save As...** command.

Please note that you cannot save fonts with more than 5,000 glyphs. If you try to save a bigger font you will see a warning message that will recommend splitting a font into smaller parts.

If you want to work with bigger fonts consider using our AsiaFont Studio product. More information about it is available at this page:

http://www.fontlab.com/html/asiafontstudio.html

After you select **File/Save As...** in the menu, you will see the standard **File Save** dialog box:



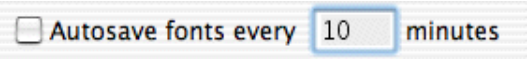
Choose the destination folder, enter the file name and click **Save** to save your font.

Autosave

If you want to protect yourself from system or program crashes you can use the Autosave function that will periodically save the current font.

To activate and customize this feature, open the **Application/Preferences** dialog box and select the **General** page.

In the top part of the page, you will see the **Autosave** controls:



Use the check box to activate Autosave and enter the time interval (in minutes) at which you want to save the font.

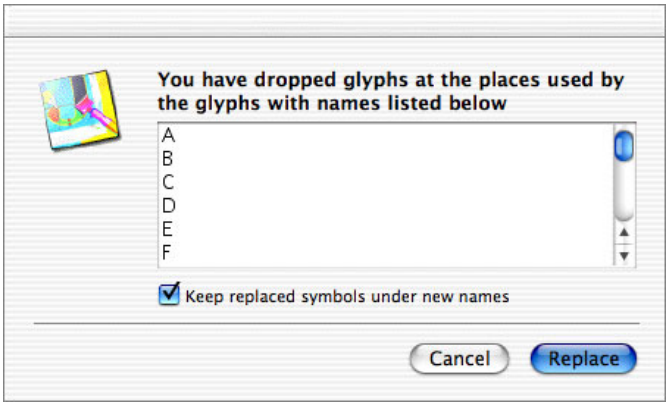
Font will be saved into the Autosave folder (subfolder in the FontLab directory) and will be named using the following structure:

Fontfilename.save.vfb, where fontfilename is the name of the font file to which you manually saved the font the last time.

If **Autosave** was active and you have a system or program crash, you can open your last saved font from the FontLab/Autosave folder.

Copying and Pasting Glyphs

To copy selected glyphs, select the **Copy** command from the **Edit** menu. Note that this copies not only the glyph outline, but also the glyph information, such as its name. The selected glyphs will be placed in the Clipboard and **can be pasted** into the same font or into another font by the **Paste** command from the same menu. Glyphs from the Clipboard will be placed starting from the first selected glyph in the destination font. If the destination position is occupied by existing glyphs a warning dialog box appears:



If you select the **Cut** command instead of the **Copy** command the glyphs will be copied to the Clipboard but will be deleted from the source positions.

You can choose the **Paste** command when the Glyph window is active. In this case the outlines layer of the first copied glyph will be added to the open glyph.

The Paste Special Command

When the common **Paste** command is used all the glyphs' layers are pasted from the Clipboard including guidelines, bitmap background, mask etc. To get more control over the pasting procedure use the **Paste Special** command in the **Edit** menu. The Paste Special dialog appears:

to let you choose what glyph information you are pasting. Check the checkboxes and click **OK** to finish pasting glyphs.

When pasting with this special command the destination selection does not have to be continuous, i.e. you can select cells to be replaced in any order and combination.

Copying Glyphs to Another Font

You can use two methods to copy glyphs from one font to another:

1. Use the **Copy** and **Paste** commands from the **Edit** menu as described in the “Copying glyphs” section, or
2. Drag them to the other font and drop them there. The drag-and-drop method is easier and more visual.

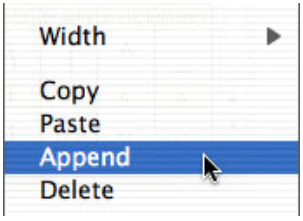
Appending Glyphs to the Font

Instead of the **Edit/Paste** command you can use the **Append** command from the Font window context menu to add glyphs from the Clipboard to the font.

When FontLab appends glyphs, it respects the glyph names and Unicode indexes, so on the first attempt glyphs will be placed in the expected code positions in the font.

Here is an example. Your first font contains Latin glyphs but has no Cyrillic glyphs. A second font is a Cyrillic font with the matching style and you want to add Cyrillic support to the first font.

1. Select the Cyrillic glyphs in the second font (this will be easy if you select the MacOS Cyrillic codepage or the Cyrillic Unicode range) and copy them to the Clipboard.
2. Return to the first font; **Ctrl**-click the Font window; and click on the **Append** command in the context menu:



3. The Cyrillic glyphs will be appended to the font with their correct Unicode indexes and names, so you will not have to re-map the font.

Copying Composite Glyphs

If you copy composite glyphs (instead of having their own outlines composite glyphs are built from references to other glyph outlines) to another font, FontLab will try to not decompose (replace references to glyph with actual glyph copies) them. Instead it will try to find matching components in the glyph set that was copied or, if some components are not present there – in the destination font.

If FontLab can completely restore composites in the destination font it will even keep TrueType hinting programs for these glyphs.

Drag-and-Drop of the Composite Glyphs

If you prefer to use the drag-and-drop method to copy composite glyphs you have one additional option: when you drop a composite glyph and FontLab finds that one or more of its components were not selected to copy and do not present in the destination font, it shows a message asking you if you want to copy all the missing components. If your answer is Yes, then FontLab will automatically append all the necessary components to the destination font so that all the composites stay unchanged.

Duplicating Unicode Indexes

In FontLab you may assign more than one (up to 63, actually) Unicode indexes to a glyph. Visually this means that a glyph that has several Unicode indexes will appear several times when one of the Unicode modes (Ranges or Codepages modes) is selected in the Font window. To distinguish the original copy of the glyph from the duplicates the latter are marked by a small blue mark in the left-top corner of the glyph cells.

To make a duplicate of a glyph, select it (you may select many glyphs at once); position the mouse cursor on the selection; press the **COMMAND** and **OPTION** keys; press the mouse button and drag the selection to the place where you want it to be duplicated. It is important to have the **COMMAND** and **OPTION** keys pressed when you release the mouse button.

You can later correct Unicode indexes assigned to the glyph by using the Rename Glyph dialog or the Glyph Properties panel (described later).

Deleting Glyphs

To remove glyphs from the font

1. Select the glyphs that you want to remove.
2. Select the **Delete** command from the **Edit** menu or from the popup menu. Or, press the **DELETE** or **BACKSPACE** key on the keyboard.
3. A dialog box appears asking you if you are sure that you want to delete.

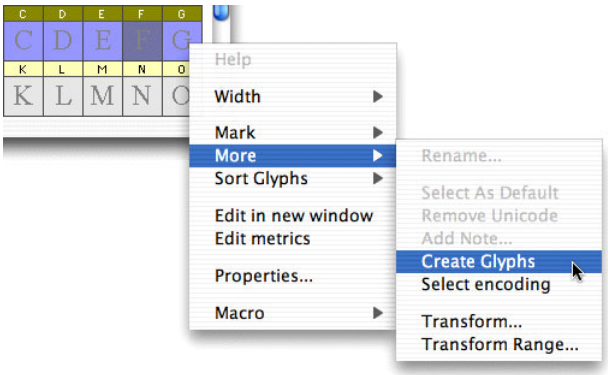
✓ **Note 1:** Deleting glyphs from the font is not undoable, so save your work before deleting glyphs.

✓ **Note 2:** If you are in Unicode mode and deleting glyphs with the blue mark in the top-left corner, they will be removed without any questions because they are just one of the indexes of a multi-Unicode glyph.

Creating New Glyphs

If you want to create a new glyph in an empty place in the font (a gray cell in the Font window), double-click the cell.

If you want to create a group of new glyphs with a single command, select the empty cells; **Ctrl**-click the Font window; and use the **More/Create Glyphs** command in the context menu:



If you are creating glyphs in the “yellow zone”, names and Unicode indexes are assigned to the newly created glyphs according to the selected encoding table.

Highlighting Glyphs

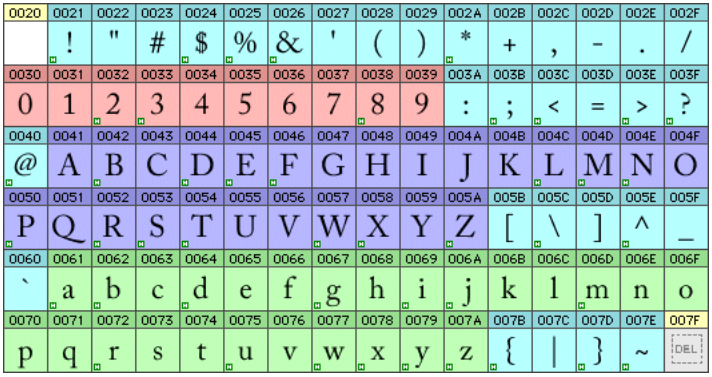
Sometimes you need to visually differentiate groups of glyphs to easily examine and select them.

FontLab provides two methods of differentiation: marking and definition of a glyph collection.

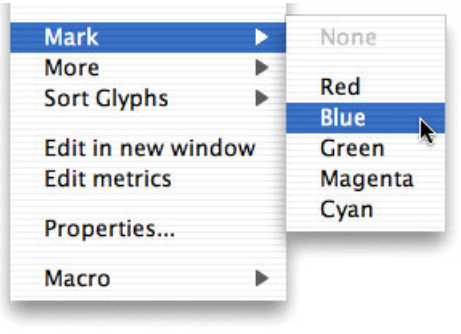
Marking Glyphs

With marking you can add color to a glyph cell in the Font window. There are five predefined colors: **red**, **blue**, **green**, **magenta** and **cyan**.

Marking is useful when you want to add structure to a font – for example, make a visual difference between digits, uppercase and lowercase characters. If you want to use this information internally (in macros), we recommend you use glyph classes (described in the “OpenType Fonts” chapter), but if you only want to have visual differentiation, marking is OK:

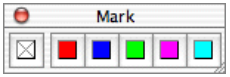


To mark the glyphs, select them in the Font window; **Ctrl**-click; and select the marking color in the **Mark** submenu of the context menu:



To remove marking, select the marked glyphs; **Ctrl**-click; and select **None** in the **Mark** menu.

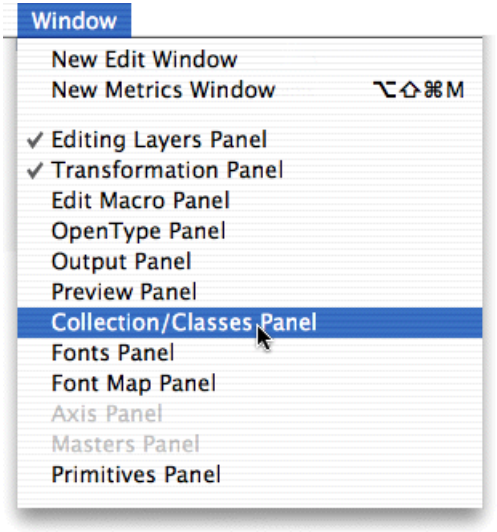
To simplify repetitive marking operations you can make the Mark toolbar visible by selecting it in the **View/Toolbars** submenu:



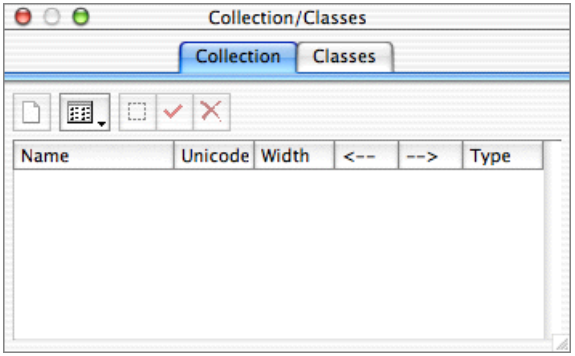
Glyph Collection

Another method of differentiating glyphs is to put them into a glyph collection. A glyph collection is a simple list of the glyph indexes visually represented in a special panel.

To begin working with the glyph collection, open the Collection/Classes panel using the command from the **Window** menu:



You will see a panel with two pages: Collection and Classes. We will discuss glyph classes later, so activate the Collection page now:



In the top of the panel you will see a small toolbar

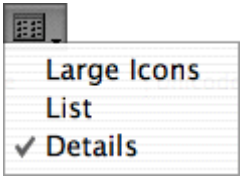
and a list of glyphs below it.

To add a glyph to the collection, select it in the Font window and drag it to the Collection panel's glyph list.

The list works in "Report" mode by default and has several columns:

Name	Glyph icon and name
Unicode	Unicode indexes assigned to the glyph
Width	Glyph width
←	Left sidebearing
→	Right sidebearing
Type	Glyph type (normal glyph, ligature or accent mark)

To switch the list to a different mode, use this button on the toolbar and select one of the modes in the menu:



Other buttons on the toolbar mean:

	Reset collection	Removes all glyphs from a collection
	Select collection	Selects glyphs in the Font window that are selected in the glyph list in the Collection panel
	Mark collection	Marks glyphs that are selected in the list in red
	Remove glyph	Removes the selected glyphs in the list from the collection

Using the Collection Panel

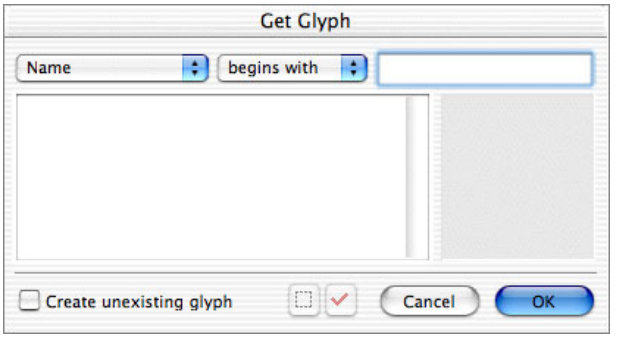
You can drag-drop glyphs selected in the Collection's list to the Glyph window (the first glyph will be selected for editing) and to other FontLab panels.

If you double-click a glyph icon in the Collection's list, you will see a new Glyph window appears with this glyph prepared for editing.

If the list is in the Report mode, click on the header to sort the glyphs according to one of their properties.

Searching for Glyphs

Sometimes you need to find a particular glyph in your font, especially in large fonts. Select the **Choose Glyph** command in the **Glyph** menu or press **CMD+BACKSPACE** on the keyboard. You will see a dialog box:



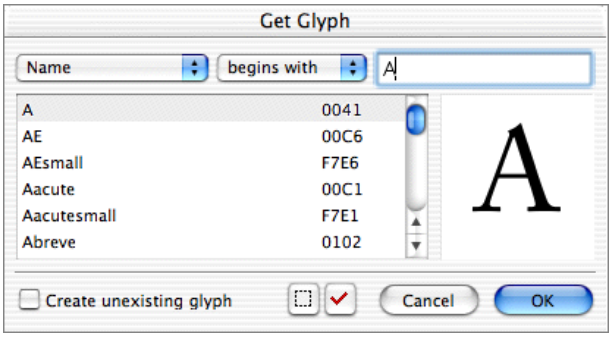
To find a glyph:

1. In the left-top popup menu select the method by which you want to search for the glyph:

Name	Searches for the glyph name
Code	Searches for the decimal code of the glyph in the current encoding or codepage
ANSI glyph	Searches for the glyph that is mapped to one of the ANSI glyphs in the selected codepage or encoding
Unicode index	Searches for glyphs with Unicode index attributes given
Width	Searches for glyphs with the width in the selected range
Bottom, Top	Searches for glyphs whose bottom or top line falls in the specified range
Components	Searches for glyphs that have the specified number of components.

2. In the popup menu to the right of the method select the comparison factor: begins with, equals to, less than or more than.
3. In the right-top editing field enter the information (depending on your selection) that will be used to find the glyph.

4. The names of all the glyphs that match the criterion will appear in the list.



Select the glyph name that you want (its preview appears in the preview panel) and press **OK**, or enter more information to narrow your search.

Use the buttons at the bottom of the dialog box for additional features:

to select (in the Font window) all glyphs in the list

to mark in red (in the Font window) all glyphs in the list.

These features are very useful for managing big fonts. For example: open the **Choose Glyph** dialog box; change it to Names mode and enter 'A' as the search pattern. In a standard Latin font it will list all glyphs that begin with the uppercase A:

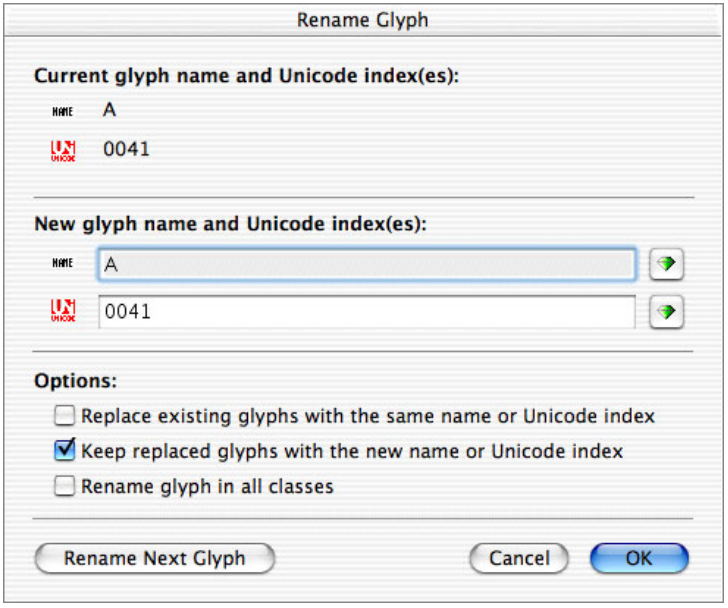
A
Aacute
Acircumflex
Adieresis
AE
Agrave
Aring
Atilde

If you mark or select them, you can easily create a glyph group that you can use in advanced features, like class-based kerning or OpenType features.

Renaming Glyphs

Usually it is not necessary to manually rename glyphs because their names and Unicode indexes are assigned automatically when you move glyphs in the Font window. But if you want to see the information and correct it, select the **Rename Glyph** command from the **Glyph** menu. Or just press **Cmd**+\ on the keyboard.

You will see a dialog box:



In the top part of the dialog box you see the current name and Unicode index (indexes) of the glyph. In the middle there are two edit fields where you may change the information. Below them lie the options controls.

To change a glyph's name enter a new name in the **Name** field. If this glyph has a properly assigned Unicode index and you want to find the name mapped to that index in FontLab's database press the **Auto** button to the right of the edit field and FontLab will fill in the **Name** field for you.

If the option **Replace existing glyphs with the same name or Unicode index** option is not checked then, if you enter a name that is already assigned to one of the font's glyphs, the **OK** button will be disabled and you will not be able to assign that name. Switch the option on to allow FontLab to replace glyphs. Use the next option to control how FontLab does the replacement.

Use the **Unicode** edit field to change a glyph's Unicode indexes. You may enter more than one Unicode index separated by a space. Use the **Auto** button to find the Unicode indexes mapped to a glyph's name in FontLab's database.

Rename glyph in all classes – this option will automatically change the glyph name in all classes that contain the glyph being renamed.

Press the **OK** button to assign a new name to the glyph. You will see that the glyph moves to a new place in the Font window depending on the currently selected encoding vector, Unicode range or codepage.

If you want to rename more glyphs, press the **Rename Next Glyph** button. A new name will be assigned to the current glyph (as if you had pressed the **OK** button) and data from the next glyph will appear for editing.

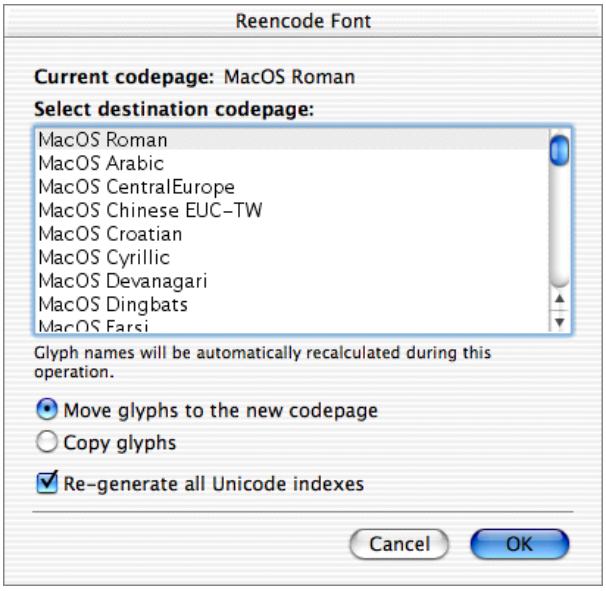
Reencoding the Font

In FontLab you can assign names from the encoding vector to glyphs that are sorted in the Font Window according to a different encoding vector. Or you can assign Unicode indexes from one codepage to the currently selected (different) codepage if the Font window is in Codepages mode.

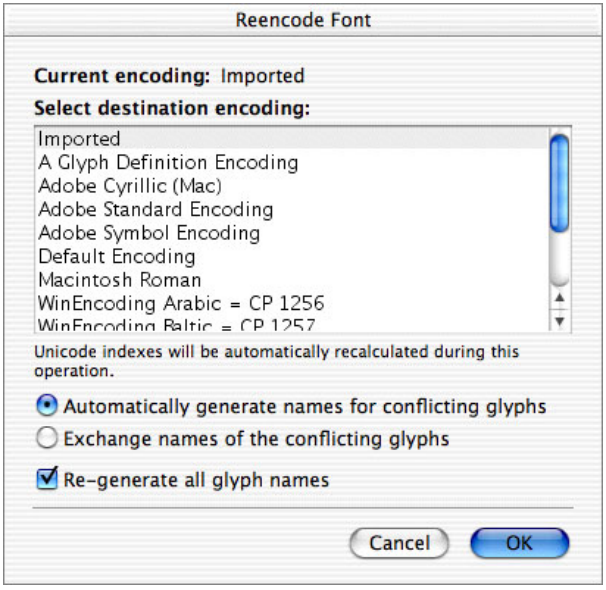
In this operation FontLab takes encoded glyphs (glyphs that are in the yellow area of the Font window's glyph chart) one by one and assigns names or Unicode indexes from the encoding vector (or codepage) that you select.

To reencode glyphs, select the Encoding or Codepages mode in the Font window. Select the Encoding mode to assign new names and the Codepages mode to assign new Unicode indexes. Then select the **Reencode glyphs** command from the **Glyph/Glyph Names** menu.

Depending on the selected mode you will see one of the following dialog boxes:



*Codepage version of the **Reencode Font** dialog box*



*Encoding version of the **Reencode Font** dialog box*

Select the codepage or encoding table to which you want to reencode the glyphs and set the options necessary to control the reencoding process:

Codepages mode:

Move glyphs to the new codepage	Removes currently assigned Unicode indexes and assigns new ones. Visually this means that glyphs are moved to their new places.
Copy glyphs	Add new Unicode indexes. Visually this means that glyphs having more than one Unicode index assigned are copied to their new places.

Encoding mode:

Automatically generate names for conflicting glyphs	If a glyph with the same name as one of the reencoding glyphs needs to be changed a new name will automatically be generated.
Exchange names of the conflicting glyphs	If a glyph with the name that needs to be assigned already exists, it will get the name of the reencoded glyph, so visually the glyphs will be exchanged.

Some Examples

Situation 1: You want to make a PC TrueType font that will have a non-windows codepage in Windows, say, one of the Macintosh codepages:

1. Select the desired new codepage in the Codepages mode of the Font window
2. Place all glyphs as necessary.
3. Save this "properly encoded" version of the font.
4. Select the **Reencode Glyphs** command.
5. Choose the Windows Symbol codepage and the "Move" option. Press **OK**.
6. See the results in the Font window.

Do not forget to set the Symbol glyph set in the Font Info before exporting this font.

Situation 2: You imported a Type 1 font with non-standard encoding and want to save it with standard encoding to be sure that it will work properly.

1. Select the **Reencode Glyphs** command.
2. Choose the Default Encoding and "Generate Names" options. Press **OK**.
3. See the results in the Font window.

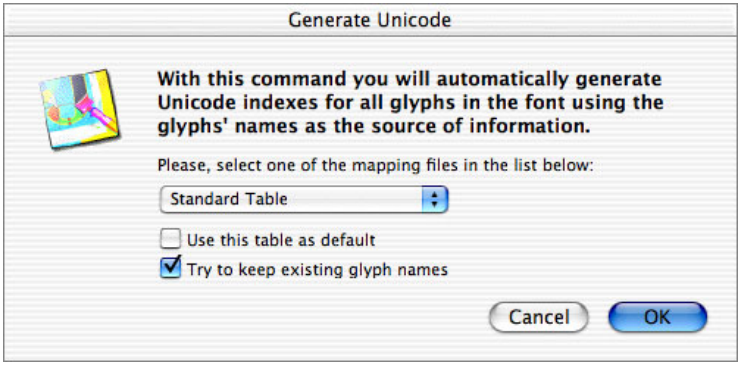
Unicode-Related Operations

Several commands in the **Glyph/Glyph Names** menu work with the font's Unicode information.

Generating Unicode Indexes

To automatically generate Unicode indexes for all the glyphs in the font, select the **Generate Unicode** command from the **Glyph/Glyph Names** menu.

You will see the dialog box:


Generate Unicode
With this command you will automatically generate Unicode indexes for all glyphs in the font using the glyphs' names as the source of information.
Please, select one of the mapping files in the list below:
Standard Table
Use this table as default
Try to keep existing glyph names
Cancel
OK

This is basically a warning message with an additional feature: it lets you choose the mapping file. The structure of a mapping file is described below, but its purpose is simple: to map Unicode indexes to a set of predefined names.

You select the appropriate file and click **OK**. Then FontLab will:

1. Remove all Unicode data.
2. Search the name-Unicode database for each glyph's name.
3. If the name is in the database it adds the Unicode index linked with this name to the glyph's list of Unicode indexes.
4. Because the database may link more than one Unicode index with a name, steps 2 and 3 are processed whenever a glyph's name is found in the database.

Structure of the Name-Unicode Database

The database that links Unicode indexes and glyph names is nothing more than a text file, STANDARD.NAM, located in the **Data** directory that has the following structure:

```
%%FONTLAB NAMETABLE
0x0000 .notdef
0x0002 nonmarkingreturn
0x0020 visiblespace
0x0020 space
. . . . . .
```

The first line of this file is a signature that is used to show that this file is a properly defined database file.

The lines that follow the signature have a very simple structure:

<Unicode index> <name>

The Unicode index may be in decimal or hex (started with '0x') form. The name should not have any spaces. Names are case sensitive.

One Unicode index may be linked with more than one name and several Unicode indexes may be linked with one name.

If the name is preceded with the '!', it means that Unicode may be generated from the name but none of the marked names may be generated when the Unicode index is known. This is necessary when none of the glyph's names is included in the list of standard names supported by Adobe (Adobe Glyph List). This feature makes it possible to generate correct Unicode indexes for incorrectly named glyphs but will never assign incorrect names.

You can extend this file in any text editor, but we strongly recommend not changing it.

Generating Names

This operation is the opposite of the one described in the previous section. If you select the **Generate Names** command in the **Glyph/Glyph Names** menu, FontLab will use the name-Unicode database to automatically find names for glyphs whose Unicode indexes are in the database.

Note that if more than one name is linked to the Unicode index, FontLab will use the one that is first in the database.

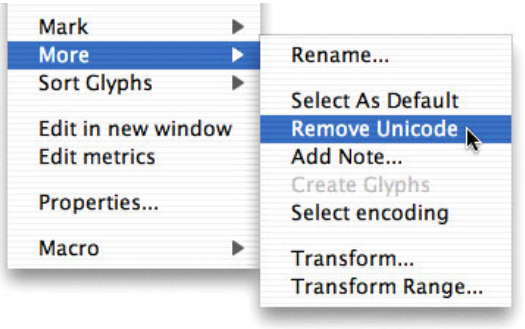
If some of the glyphs don't have any Unicode indexes, FontLab will try to keep their names unchanged. Use the following option in the dialog box to control this feature:

Removing Unicode Information

If you want to reset the Unicode information in your font, select the **Clear Unicode** command from the **Glyph/Glyph Names** menu. All Unicode information will be removed from all glyphs in the font.

You may **selectively remove Unicode indexes** from glyphs selected in the Font window:

1. Select the glyphs.
2. Ctrl-click the selection and choose the **Remove Unicode** command from the **More** submenu of the context menu:



FontLab will remove the Unicode indexes from all selected glyphs.

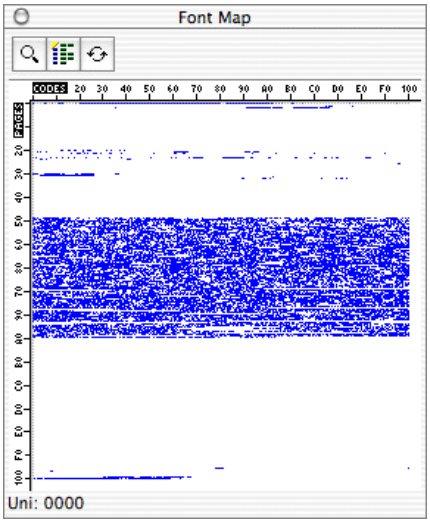
The Font Map Panel

When you work with really big Unicode-encoded fonts, you may need to have an overview of your whole font. FontLab has a special panel, called the Font Map, which can represent the entire Unicode code space as a 256 x 256 picture where every pixel represents a double-byte code.

Every pixel row in this picture represents a Unicode page – 256 Unicode indexes which begin with the same code. For example, codes A700-A7FF will form one row.

Every pixel in the row represents an individual code.

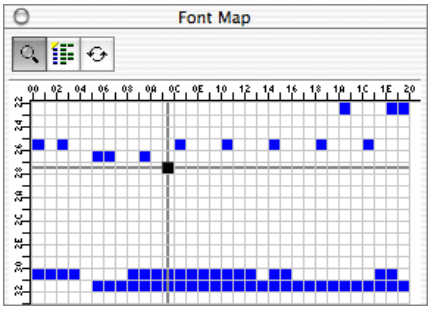
To open the **Font Map** panel, use the **Font Map panel** command in the **Window** menu. You will see a panel that consists of the code picture, toolbar and status bar:



The buttons on the toolbar mean:

- Turns on/off zoom mode
- Changes the Font Map to double-byte codepage mode
- Updates the contents of the Font Map.

By clicking the button on the toolbar you can zoom in on part of the Font Map:



In this mode it is much easier to manage individual codes. To scroll a zoomed Font Map, press the mouse button and drag the cursor beyond the Map borders.

If you click the Font Map, you will see the current Unicode index appear on the status bar below the Map picture. The current code is highlighted with a cross hair.

Double click any code in the Map **to jump to the glyph that is mapped to it**.

Font Map automatically tracks changes you make to the font. If you are not sure that it is correctly updated (this may happen with some macro programs), click on the button **to manually update the Map**.

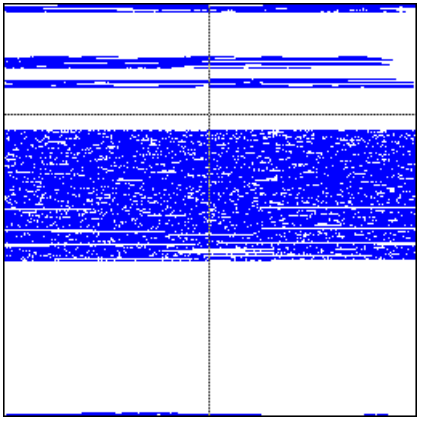
Managing Double-Byte Codepages

If you are working on a CJKV (the acronym for Chinese, Japanese, Korean and Vietnamese) font, you may want to look at your font in a double-byte codepage.

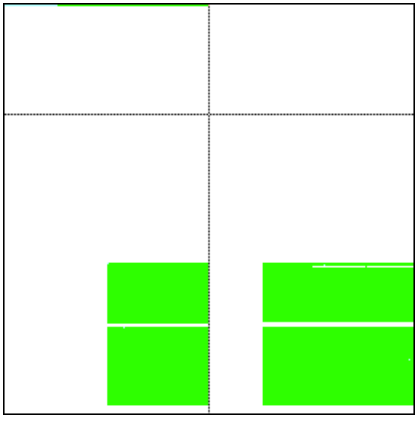
Open the Font Map panel and in the Font window of your font select one of the double-byte codepages.

You will see this button enabled in the Font Map toolbar. Click it and you will see the Font Map rearrange to represent your font with the applied double-byte codepage. In this mode every row represents 256 glyphs that are "assigned" to the specific first byte.

In the following picture you can see a Traditional Chinese font in Unicode mode (in the left picture) and in Codepage 950 mode (right):



Unicode mode



Codepage 950

In the codepage mode **green pixels** represent codes in the codepage that are covered by one of the glyphs in the font. **Cyan pixels** mean codes in the codepage that are not covered by any glyphs in the font.

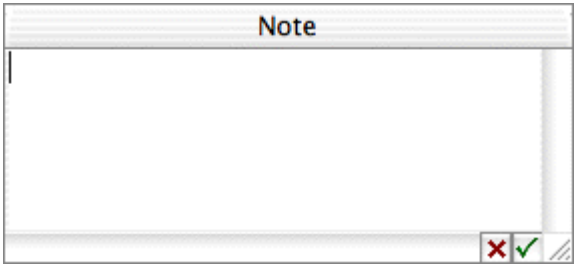
Notes

Sometimes you may need to add a description or some other information to a glyph. In FontLab you can do that using the Notes function.

A Note is a small text box that you can attach to the glyph. It is visible in the Font window and in the Glyph window.

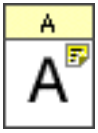
To add a note to a glyph in the Font window: select it; **Ctrl**-click and select the **More/Add Note...** command in the context menu.

You will see a Note window:



Enter the text of the note and click the button in the bottom bar to accept changes or to reject the changes and close the window. There are also keyboard shortcuts: **Cmd**+**Enter** to accept changes or **Esc** to exit without saving.

If you have entered something in the note box you will see the note icon appear in the glyph cell:



To view the note **Ctrl**-click on the note icon. You will see the note window with the text of the note. The window will disappear when you release the button.

To edit a note double-click the note icon or select the **More/Add Note...** command from the context menu.

To remove the note open it for editing, remove the text and click .

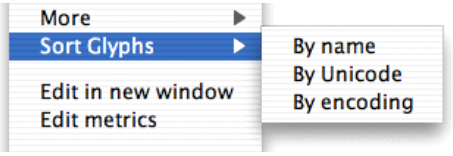
Sorting Glyphs

Sometimes you may want to sort the glyphs in a font file in some order other than their current order. There are two common reasons to do this:

1. You may need to sort glyphs to optimize a font's performance – if you place glyphs that are used most often at the beginning of the glyph collection in some cases it may improve performance.
2. You may need to sort glyphs according to some logical sequence. Some programs (for example, Adobe InDesign) have a "glyph insert" feature that shows the collection of glyphs sorted exactly as it is in the font file.

Please note that sorting of the glyphs has no effect on glyph encoding or names.

To sort the glyphs Ctrl-click the Font window and choose one of the options in the **Sort** submenu of the context menu:



This is the list of sorting options:

By name	Glyphs are sorted by their names in ascending order.
By Unicode	Glyphs are sorted by the assigned Unicode indexes in ascending order. Glyphs that don't have Unicode indexes are stored at the end of the glyph collection.
By encoding	Glyphs are sorted according to the currently selected encoding table. This option allows you to customize the sorting order using the definition encoding tables.

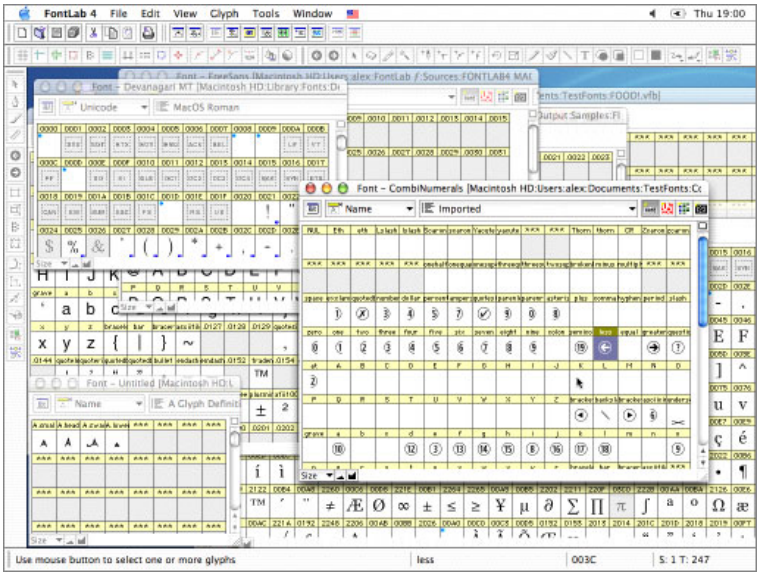
To preview the result of sorting, switch the Font window to the Index mode by clicking this button: in the Font window toolbar. You must be in Index mode to see the sequence of glyphs.

Working with Multiple Fonts

In FontLab you can open many fonts at once. Since every font has its own Font window sometimes the FontLab workspace becomes so crowded with windows that finding a particular font is not easy. This section will explain how to use FontLab tools that are specially designed to help you manage many open fonts simultaneously.

To open many fonts you can use the standard **File/Open** command, then select many font files using **Cmd** and **Shift**-click in the **File Open** dialog box. You can also select font files in Finder and drag them onto the FontLab icon – all of them will be opened.

When you open 15 fonts the screen might look like this:


FontLab 4 File Edit View Glyph Tools Window
Thu 19:00
Font – CombiNumerals [Macintosh HD:Users:alex:Documents:TestFonts:Cc
Font – Untitled [Macintosh HD:L
Use mouse button to select one or more glyphs
less
003C
S: 1 T: 247

Which is not the best way to work. Add to this picture a few open Glyph and Metrics windows and you will see why workspace management is necessary.

Window Menu

The easiest way to manage open windows is to use the **Window** menu. It contains some very useful commands:

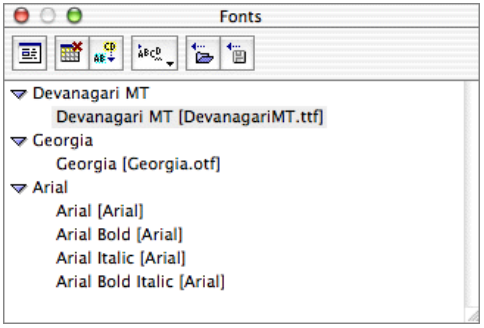
Cascade	Organizes open windows in a cascade.
Tile horizontally **Tile vertically**	Organizes windows like tiles on a rectangular floor.

Remember that you may always use the Dock to store windows that are currently not used.

Fonts Panel

The **Window** menu is powerful, but it works only on windows, without paying any attention to the contents of the windows – fonts.

To organize your open fonts, use the Fonts panel. Choose the **Fonts panel** command in the **Window** menu. When you click on this command you will see the panel:



The Fonts panel contains a small toolbar in the top area and a list of open fonts. All open fonts are automatically arranged in font families using the Family Name entry in the font header (see the "Font Header" chapter). You can expand and collapse families to better organize space in the list.

Double-click a font name in the list to activate the Font window containing that font. All other operations are accessible from the toolbar:

	Opens the Font Info editor for the font selected in the list.
	Closes the selected font. If you have unsaved changes FontLab will issue a warning message.
	Merge fonts – adds glyphs from the font selected in the list to the currently active font (contained in the active Font window).
	Opens a menu with macro programs that can be applied to the font selected in the list.
	Opens the FontLab workspace file.
	Saves the workspace file.

Merging Fonts

With the Fonts panel you can also merge fonts – append all the glyphs from one font to another. During this process FontLab will try to keep hinting information, composite glyphs and kerning pairs. FontLab will also try to keep the Unicode indexes and names of the appended glyphs. If copied glyphs have names that already exist in the destination font, FontLab will add a suffix with a number to the name.

To merge fonts:

1. Open the two fonts you want to merge.
2. Open the Fonts panel.
3. Activate the font **to** which you want to append glyphs – double-click the font name in the Fonts panel.
4. In the Fonts panel select the font **from** which you want to append glyphs.
5. Click the **Append** button in the Fonts panel toolbar.

Merging fonts is very useful when you have several fonts that cover different scripts and want to combine them into one big Unicode font. Smaller fonts are easier to manage and open/save.

To keep your font project intact FontLab offers some workspace management functions:

Saving and Opening a Workspace

A Workspace is a set of fonts opened in FontLab and information about the position and size of their font windows.

To save a workspace, use the button in the Fonts panel toolbar. FontLab will open a standard **File Save** dialog box where you can select a name and path for the workspace file. FontLab workspace files have the extension ".flw".

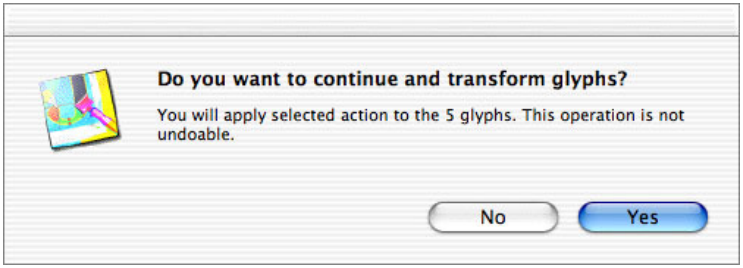
To open a workspace file, use the button in the Fonts panel toolbar or simply drag-drop the .flw file onto the FontLab icon. **Note that FontLab workspace files won't open if Python is not installed on the System**.

FontLab workspace files are automatically generated macro programs in the Python scripting language. They are text files that you can edit with any text editor if you wish. (Recommended only for the knowledgeable and brave.)

Applying Actions

You can find many actions in the **Glyph** and **Tools** menus that can be applied to the open glyph in the Glyph window (see next chapter), but most of them are applicable to the glyph(s) selected in the Font window.

To apply an action, select the glyphs in the Font window and choose the appropriate command in the **Tools** or **Glyph** menu. For example, to convert glyphs from TrueType outlines to Type 1 outlines, select the glyphs that you want to convert and choose the **Tools/Outline/To Type 1 Curves** command. You will see the warning message:


Do you want to continue and transform glyphs?
You will apply selected action to the 5 glyphs. This operation is not undoable.
No
Yes

Click **Yes** and FontLab will apply the action to all selected glyphs.

You can find descriptions of all actions in the chapter “Glyph Window”.

Blending Fonts

With FontLab you can automatically blend two fonts and generate an intermediate version of them:

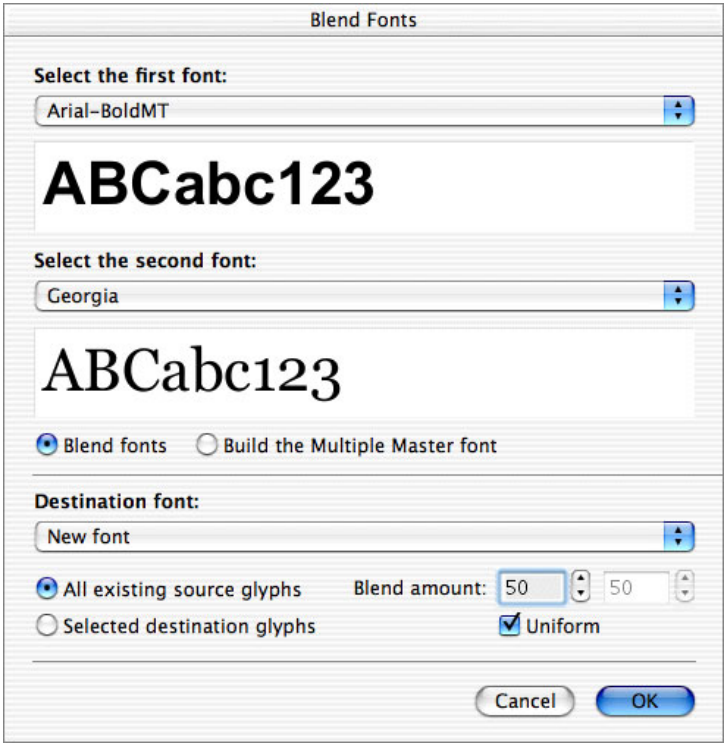
First font

Second Font

Resulting font (50% blend)

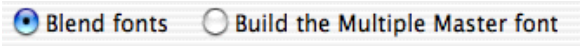
The font blending process is completely automatic, it analyses the shapes of glyphs and tries to find the best way to morph one to another. Best example of blend feature is to make a font that has weight intermediate to two existing weights.

You must have at least two fonts open to use the Blend fonts operation. To activate the Blend feature, select the **Blend Fonts** command from the **Tools** menu. A dialog box appears:



The dialog box is divided into three sections. The first two sections let you specify the fonts to blend. To select the first or second font choose the font name in the popup. The Preview panel will show a sample of the font. You cannot select the same font in both sections; FontLab will take care of that.

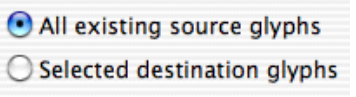
The bottom section lets you specify blend options.



You have two main choices: to make a single master font that will contain the result of the source fonts blend or to make a Multiple Master font that will have the two source fonts as the masters. Here we will describe only single-master mode. Please refer to the Multiple Master chapter for information about using the Blend feature to make Multiple Master fonts.

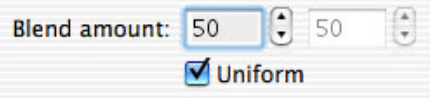
In the “Destination font:” popup menu you can select the font in which to store the blended glyphs. Choose “New font” to put the generated glyphs into a new font.

Further down there are two options:



Select the second option if you want to blend only glyphs that are selected in the **destination** font.

To the right there are controls to specify the blend amount:



Enter the blend percentage in the editing field. If you want to blend fonts nonproportionally along the X and Y axes, uncheck the **Uniform** check box and enter the blend amount for the two directions.

Click **OK** and wait while FontLab blends the fonts. After the process is finished you will find the new glyphs in the new font or in the destination font. The outlines of the two source glyphs will be stored in the Mask layer of each glyph and the blending result in the outline layer.

In some cases it is not possible to blend two outlines. This happens when two source glyphs have different numbers of contours, like O and 8. If some glyphs were not blended, FontLab will show a warning message and those glyphs will have an empty outline layer.

Exporting Fonts

You know how to open fonts, how to rearrange characters and change encodings and how to save fonts in FontLab format to keep your changes. Now it's time to learn how to generate fonts in formats that other programs can understand.

Please note that you cannot export fonts with more than 5,000 glyphs. If you try to export a bigger font you will see a warning message that will recommend that you split the font into smaller parts.

If you want to work with bigger fonts consider using our AsiaFont Studio product. More information about it is available at this web page:

http://www.fontlab.com/html/asiafontstudio.html

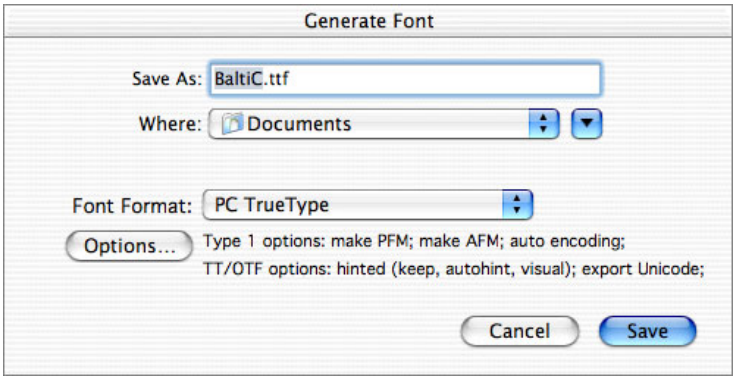
Font Formats

Macintosh version of FontLab can work with four font formats and their variations:

PC Type 1 and Multiple Master	Sometimes fonts in this format are called PostScript fonts. Usually they consist of two or three files: a file with the .pfb extension and one or two additional files needed by the OS to get information about the font header and font metrics. You need Adobe Type Manager to use these fonts on Windows (if you are not using Windows 2000 or Windows XP).
Mac Type 1 and Multiple Master	On Macintosh computers fonts in this format are always called PostScript fonts. Usually they consist of two resource files: a suitcase file with at least one bitmap font (needed by the Mac OS to get information about the font family) and the file with 'post' resources containing font-definition data.
PC TrueType	The standard format for Windows fonts since version 3.1 and for the Mac OS. The font is stored in a single file that has a .ttf extension. TrueType-based OpenType fonts have the same structure but some additional information. From FontLab's point of view, there is minimal difference between "plain" TrueType fonts and TrueType-based OpenType fonts. Mac OS now supports PC TrueType.
Mac TrueType	The font is stored in a single resource (Pre Mac OS X) or plain data-fork (on Mac OS X with .dfont extension) file that is called a suitcase. One suitcase can contain up to 4 TrueType fonts combined in a family.
OpenType	A relatively new font format jointly supported by Microsoft and Adobe. Files can have either a .ttf or .otf extension. Mac OS X, Windows 2000 and Windows XP directly support these fonts but the .otf version of the format requires Adobe Type Manager on Windows 9x and ME. OpenType fonts may be TrueType-based (which we will call TrueType fonts unless specially noted) or PostScript-based. The latter are also called CFF-based or CFF-flavored. CFF is a Compact Font format designed and supported by Adobe. Think of it as an improved and more compact version of Type 1.
FontLab	This is FontLab's own font format that potentially can contain a superset of information that you can export to other formats. You already know how to open and save FontLab fonts.

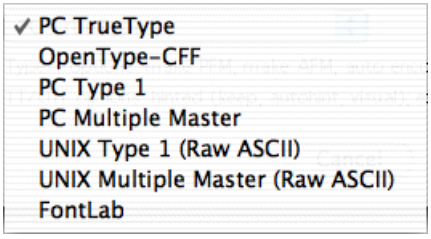
Exporting for PC

To save a font in PC Type 1, PC TrueType or OpenType format use the **File/Generate Font** command. You will see the **Generate Font** dialog box:



The top part of the dialog box is standard and there you choose a destination folder and enter a file name for the font file. By default FontLab will choose the folder where you saved fonts last time.

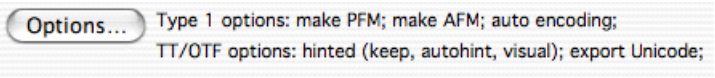
Below the standard part of the dialog box is format selection popup menu. Choose the destination font format there:



Available formats are:

PC TrueType (ttf)	TrueType and TrueType-flavored font format. Depending on the OpenType export options (discussed below) FontLab may write the additional data and tables that make the difference between "plain" TrueType and TrueType-based OpenType.
OpenType-CFF (otf)	CFF-flavored OpenType font format. Refer to the "Making OpenType Fonts" chapter for more information about the OpenType fonts.
PC Type 1 (pfb)	Single-master Type 1 font format in a binary form. This is the format for Type 1 fonts that is expected by Windows. Multiple Master fonts exported as single-master Type 1 fonts are converted to a single-master font with the default weight vector of the font.
PC Multiple Master (pfb)	Multiple Master Type 1 font format. If your font is single-master, then this item is not available in the popup.
UNIX Type 1 (raw ASCII) (pfa)	Single-master Type 1 font format in ASCII (7-bit) form. This format is used in most UNIX systems and can be directly downloaded into PostScript printers.
UNIX Multiple Master (raw ASCII) (pfa)	ASCII (7-bit) form of the Type 1 Multiple Master format. If your font is single-master, then this item is not available in the popup.
FontLab (vfb)	FontLab internal font format. This format it is used when you select the **File/Save As...** command.

Below the format selection popup menu is the **Options** button (that opens the **Export Options** dialog box described later in the chapter) and a legend field that describes the currently selected options for TrueType and Type 1 formats:



Check the export options in the dialog and press the **Save** button to export the font, or **Cancel** – to return back to font editing.

Exporting for Macintosh

As far as Mac OS X now supports PC TrueType and OpenType fonts you may choose these formats available in the Generate Font dialog box described in the previous section. We will describe here the font suitcase generating procedure.

Font Suitcases

On the Macintosh fonts that belong to a font family are physically combined into a single file, called a font suitcase. The suitcase contains information about the font family in general and refers to the records that describe the fonts' style, encoding, metrics and kerning information and some other data that is necessary for the Mac OS to use the font.

When Type 1 fonts are used on Macintosh, files that contain actual Type 1 font-definition data are stored in a separate file that is referenced from the font suitcase.

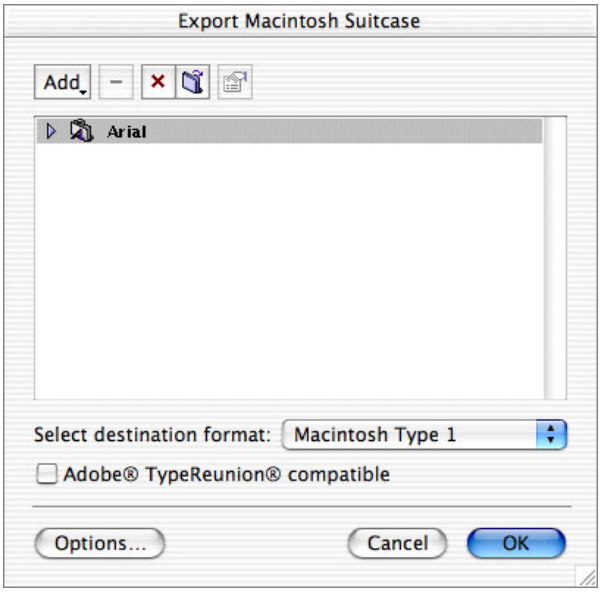
FontLab has a special dialog to compose and export Macintosh suitcases – the **Export Macintosh Suitcase** dialog. You can group several open fonts to collect them in a new destination suitcase or to add them to an existing one.

To build a proper suitcase, one must fill in the fonts' Font Info fields properly (described in full detail in the "Font Header" chapter). There are several basic rules:

1. One suitcase can contain fonts from only one font family. This means that the **FamilyFOND Name** fields of all the included fonts must be the same.
2. FontLab designates the Macintosh's bold style for a font if the font's **Weight** field is set to "DemiBold" or more.
3. FontLab designates the Macintosh's italic style for a font if the **Font is italic** checkbox is set.
4. A maximum of 4 fonts can be combined as linked styles: style Plain or Normal (Mac font style nr. 0), style Bold (Mac font style nr. 1), style Italic (Mac font style nr. 2) and style BoldItalic (Mac font style nr. 3).

Building Font Suitcases

To save a font in traditional suitcase-based Mac Type 1 or TrueType format use the **File/Generate Mac Suitcase** command. You will see a special suitcase managing dialog box:



The **Export Macintosh Suitcase** dialog box consists of 4 parts: a toolbar on the top, a list in the middle, an options area and three buttons at the bottom. In the simplest case you can just press the **OK** button and get a font, but usually some management is needed.

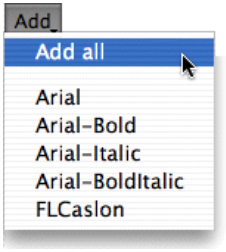
By default the list already contains a suitcase that will be generated from the font that was active when you select the **File/Generate Mac Suitcase** command. Click on the triangle on the left of the suitcase icon to see its contents. If you feel the style is not what you've expected, click **Cancel** and check the fonts' Font Info fields carefully.

The toolbar contains the following buttons:

Add	Opens the menu allowing you to add open fonts to the list
–	Deletes the selected font suitcase from the list or font from the suitcase
×	Completely clears the contents of the list
	Opens the standard File Open dialog allowing you to open suitcases from the disk
	Opens the **FOND Info** dialog box for the selected suitcase or **Font Info** dialog box for the selected font.

What you can do with these commands?

To add all fonts that have been opened with FontLab in their Font windows click on the Add button and select the **Add all** command from the menu:



FontLab checks the FOND Name field in every open font and creates suitcases one for every different FOND name. If several fonts have this field the same they will be combined in one suitcase with this name.

If the list of suitcases is not empty (and it is so after you select the **File/Generate Mac Suitcase** command), you have the possibility to add only those fonts that are compatible with this suitcase in the list: choose the **Add possible** command from the **Add** pulldown menu.

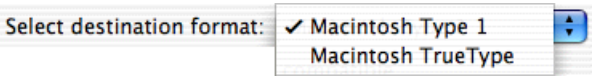
To remove all the fonts from the list click on the × button.

To remove a suitcase or a particular font in a suitcase select it in the list and click on the – button. If you remove the last font from the suitcase, then the suitcase is removed too.

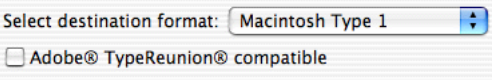
To add new fonts to existing suitcase, first clear the list with the button, then click on the button. The standard File Open dialog box appears where you may choose the suitcase from the disk. Press the **Open** button in the dialog and the selected suitcase will appear in the list. Click on the Add button and select the font that you want to be added to the opened suitcase. If the Font Info fields of the added font are filled properly it will appear in the corresponding style of the font suitcase. You have the possibility to drag-drop individual fonts between the styles in a suitcase.

Options

Before generating a Macintosh suitcase you have **to choose the font format**. Select Macintosh TrueType or Type 1 in the popup menu:

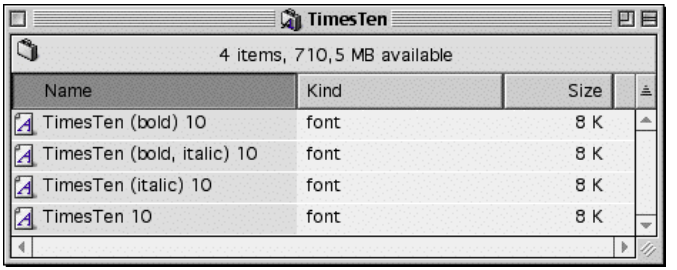


When Type 1 is selected additional option is available:

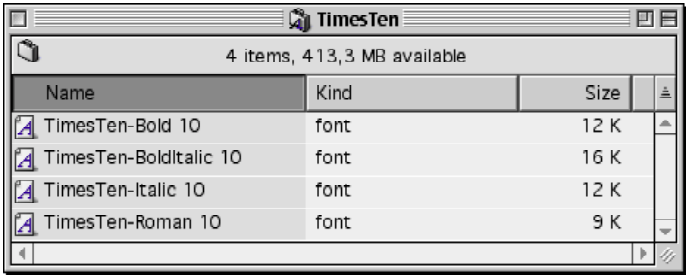


If this option is checked, FontLab will generate a font suitcase compatible with Adobe® TypeReunion®, which lets the font's styles appear in the hierarchical menu of the font menu:

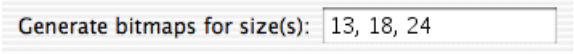
Finder's view of a suitcase created with the "Adobe® TypeReunion® compatible" checkbox switched off:



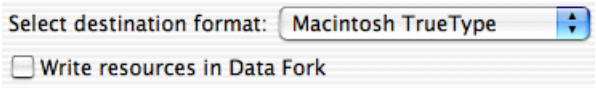
The Finder's view of the suitcase created with the "Adobe® TypeReunion® compatible" checkbox switched on is as follows:



On Macintosh Type 1 fonts have to be generated with at least one accompanying bitmap font in a 'NFNT' resource. You can define the point sizes that will be generated on the Type 1 page of the **Application/Preferences** dialog box:



When TrueType is selected another additional option is available:



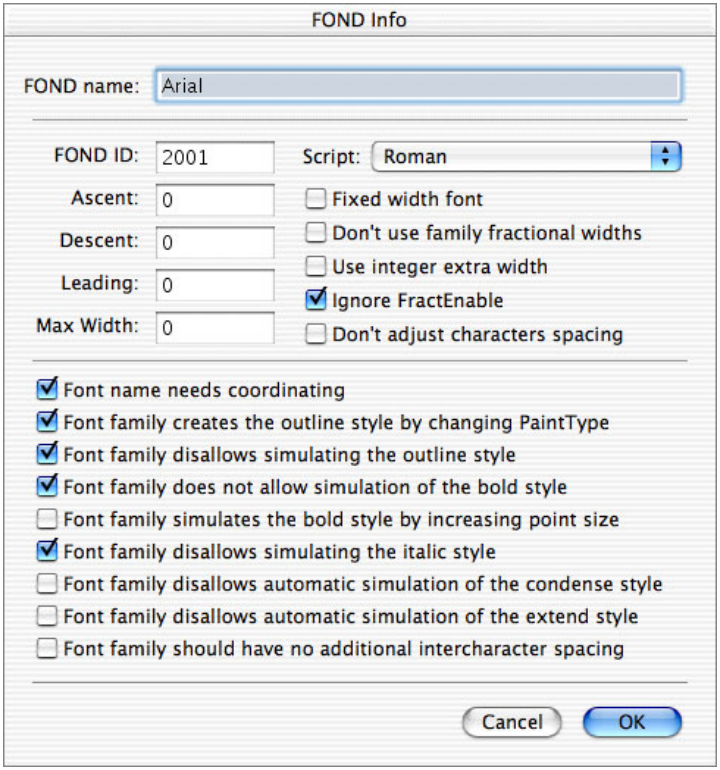
If this option is checked, FontLab will write a font suitcase as data-fork-based file supported by Mac OS X only. Leave this option unchecked if you want your font to be compatible with Classic.

Other font exporting options are available in the **Export Options** dialog box that appears when you click on the **Options** button at the bottom of the **Generate Mac Suitcase** dialog box and described later in the chapter.

After you finished managing suitcases in the list press the **OK** button and select the destination for the fonts in the standard Save dialog. Name the suitcase file if needed and press the **Save** button to generate fonts.

Family Info

To view and edit the font family information select the suitcase in the list and click on the button. The FOND Info dialog box appears:



Besides the FOND name you may edit parameters in the following groups:

Font family properties:

FOND ID and Script	FOND resource identifier (or family ID number) lying in the range of the particular script. Changing the script in the popup menu to the right will automatically change FOND ID and vice versa
Fixed width font	If this option is switched on, the font will be treated by the Macintosh system as one with characters of fixed width (monospaced). Otherwise, the font is treated as proportional
Don't use family fractional widths	If this option is switched on, the system will not use the global family widths table
Use integer extra widths	If this option is switched on, the system will use the family style extra widths table (Family Style Property Table)
Ignore FractEnable	If this option is switched on, the system will use the family style extra widths table (Family Style Property Table) even if the option **Don't use family fractional widths** is switched off
Don't adjust characters spacing	This option represents the 11th bit of the family flags, which is usually set to zero.

Font metrics:

Ascent	The maximum height above the baseline reached by characters in this family fonts
Descent	The maximum depth below the baseline reached by characters in this family of fonts. The depth is usually a negative number
Leading	Maximum leading for the family. The leading value is usually set to zero
MaxWidth	Maximum character width for the family.

Style mapping flags (Font Class):

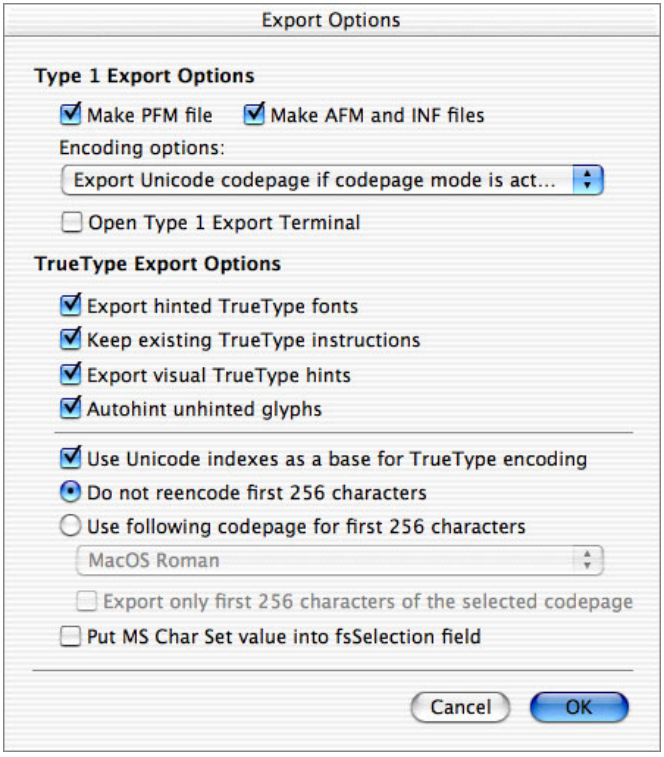
Font name needs coordinating	This option is switched on if the font name needs coordinating
Font family creates the outline style by changing PaintType	When this option is switched on, the **Outline** style of the family will be created by changing PaintType, a PostScript variable, to 2
Font family doesn't allow simulation of the outline style	This option is switched on if the font family disallows simulating the **Outline** style by smearing the glyph and whiting out the middle
Font family doesn't allow simulation of the bold style	This option is switched on if the font family disallows simulating the **Bold** style by smearing the glyphs
Font family simulates the bold style by increasing point size	This option is switched on if the font family simulates the **Bold** style by increasing the point size
Font family doesn't allow simulation of the italic style	This option is switched on if the font family disallows simulating the **Italic** style
Font family doesn't allow simulation of the condensed style	This option is switched on if the font family disallows automatic simulation of the style **Condensed**
Font family doesn't allow simulation of the extended style	This option is switched on if the font family disallows automatic simulation of the style **Extended**
Font family should have no additional intercharacter spacing	This option is switched on if the font family should have no additional spacing other than the space character.

To get full information about the parameters represented in the FOND Info dialog, refer to Inside Macintosh: Text:Font Manager:
http://developer.apple.com/techpubs/mac/Text/Text-181.html

Export Options

There are two methods of selecting export options. The basic set of TrueType and Type 1 options is available if you press the **Options** button in the **Generate Font** or **Export Mac Suitcase** dialog box. If you need to get access to all the options, use the **Type 1**, **TrueType** and **OpenType** pages of the **Application/Preferences** dialog box.

This is what you see if you click on the **Options** button in the **Generate font** or **Export Mac Suitcase** dialog box:


Export Options
Type 1 Export Options
Make PFM file
Make AFM and INF files
Encoding options:
Export Unicode codepage if codepage mode is act...
Open Type 1 Export Terminal
TrueType Export Options
Export hinted TrueType fonts
Keep existing TrueType instructions
Export visual TrueType hints
Autohint unhinted glyphs
Use Unicode indexes as a base for TrueType encoding
Do not reencode first 256 characters
Use following codepage for first 256 characters
MacOS Roman
Export only first 256 characters of the selected codepage
Put MS Char Set value into fsSelection field
Cancel
OK

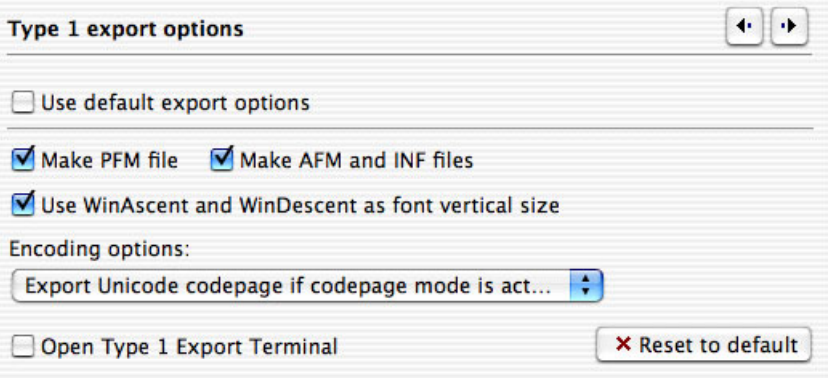
The **Export Options** dialog box consists of two areas: *Type 1 export options* and *TrueType export options*. In the following sections we will discuss exporting options in full detail.

Font-Specific Export Options

In FontLab you can specify export options for individual fonts. Use the **File/Font Info** command to open the **Font Info** dialog box (described in full detail in the “Font Header” chapter).

In the dialog select the “Type 1 specific-settings” item in the list of pages to customize Type 1 export options or “TrueType-specific settings” to customize TrueType export.

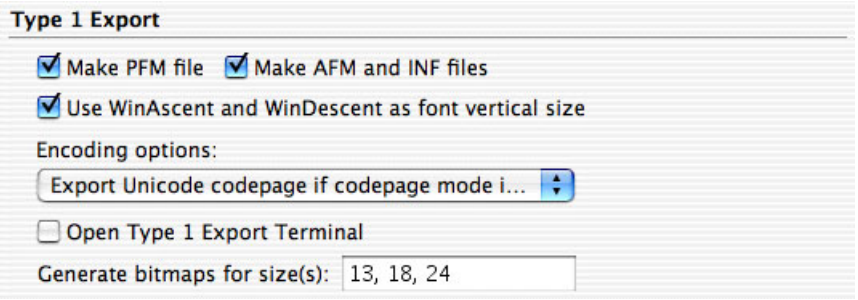
Switch off “Use default export options” at the top of the panel and you will get access to font-specific customization:


Type 1 export options
Use default export options
Make PFM file
Make AFM and INF files
Use WinAscent and WinDescent as font vertical size
Encoding options:
Export Unicode codepage if codepage mode is act...
Open Type 1 Export Terminal
Reset to default

The set of controls in these panels is almost the same as in the **Application/Preferences** dialog box. The only difference is that “global” options are applied to all fonts that have the “Use default export options” setting activated and custom options are selected only for the font for which they are specified.

Type 1 Export Options

To get access to the full set of Type 1 export options, open the **Application/Preferences** dialog and select the Type 1 page:


Type 1 Export
Make PFM file
Make AFM and INF files
Use WinAscent and WinDescent as font vertical size
Encoding options:
Export Unicode codepage if codepage mode i...
Open Type 1 Export Terminal
Generate bitmaps for size(s):
13, 18, 24

Type 1 exporting options in the **Export Options** dialog box are a subset of the options accessible in the **Application/Preferences/Type 1** dialog box. Usually the subset is enough, but sometimes you may need to customize the full set.

Make PFM File

Switch this option on to create a PFM (Printer Font Metrics) file when exporting a PC Type 1 font. PFM files are used in Windows to register Type 1 fonts. They contain metrics, kerning and, partially, font header information. You cannot install a Type 1 font on Windows without a PFM file. This option is ignored when saving Macintosh Type 1 fonts.

Make AFM and INF files

Switch this option on to make AFM (Adobe Font Metrics) and INF (font INFormation) files when exporting a Type 1 font. These files are text files and contain descriptions of the font metrics, kerning and header (font names, weight, width, encoding and other information). It is possible to install a Type 1 font on Windows with Adobe Type Manager if you don't have the PFM file but do have the AFM and INF files, because ATM will automatically build the PFM file using data from the AFM and INF files.

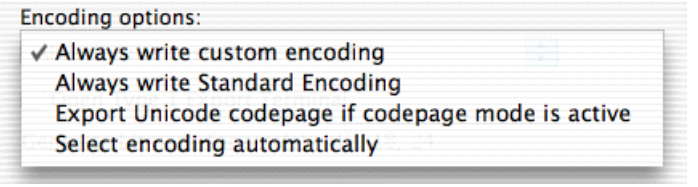
The AFM file is necessary to install a Type 1 font (in ASCII form, with ".pfa" extension) on most Unix-based operating systems.

When saving Macintosh Type 1 fonts only AFM files are created.

To install an exported Type 1 font in Windows you must have the PFM or AFM+INF files. We recommend making *all* these files when you finally produce a font so that your font will be compatible with various environments.

Encoding Selection Options

The dialog has a list of possible encoding-export options:


Encoding options:
✓ Always write custom encoding
Always write Standard Encoding
Export Unicode codepage if codepage mode is active
Select encoding automatically

When a Type 1 font is exported you have the choice of including a custom encoding vector in its body or just exporting a StandardEncoding key word that means that the font has the standard set of glyphs and that the standard system remapping tables may be applied to it.

StandardEncoding is the recommended choice if your font is a typical Latin font. If the font is in a non-standard encoding, like Cyrillic, Greek, or Arabic, you must export it with the specific encoding. In this case the encoding that is currently selected in the Font window will be saved with the font and system will use it to access glyphs. If your font contains many glyphs that can cover more than one encoding, then system will take the main encoding (if present in the font, or if assumed, as in the case of StandardEncoding) as the base encoding and will take all the other possible encodings from it.

How ATM Interprets a StandardEncoding

When a Type 1 font has StandardEncoding ATM assumes that this font includes all the glyphs from the first 128-glyph range (digits, alphabet and basic punctuation) and the European glyphs (128-255 range). The first 128 glyphs are called the "*top zone*". The 128-255 range is called the "*bottom zone*". The Adobe StandardEncoding includes very few glyphs from the bottom zone compared to the number of glyphs in the Macintosh Roman encoding. When a Type 1 font in StandardEncoding is installed with ATM, ATM uses a special encoding instead of the "real" StandardEncoding as it is documented in the Type 1 format specification. This special encoding is called the Default Encoding in FontLab. So if you create a StandardEncoding font and want to see how it will work in Mac OS, select the Default Encoding in FontLab.

Here is an explanation of the possible encoding export options:

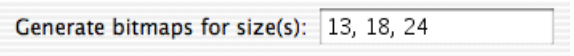
Always write custom encoding	Always saves the custom encoding with the font. Never writes StandardEncoding. The encoding currently selected in the Font window will be exported.
Always write StandardEncoding	Always exports the font with the StandardEncoding.
Select encoding automatically	Exports the font with the StandardEncoding when the Font window shows StandardEncoding, Default Encoding or Mac Roman. Exports a custom encoding if any of the other encodings is selected or if the Font window is in Codepages mode.
Export Unicode codepage if codepage mode is active	Exports a custom encoding based on the currently selected codepage if the Font window is in Codepages mode.

We recommend setting the **Select encoding automatically** option as the default, because it covers most exporting situations very well.

Generation of Bitmaps

On the Macintosh Type 1 fonts must have associated bitmap fonts for the screen that are necessary for the system to store the font's metrics. FontLab can automatically generate bitmap fonts if ATM (under Mac OS 9) is installed on your computer. It is strongly recommended that you have ATM installed if you plan to make professional-quality Type 1 fonts because this is the only way to make bitmap and outline fonts totally compatible.

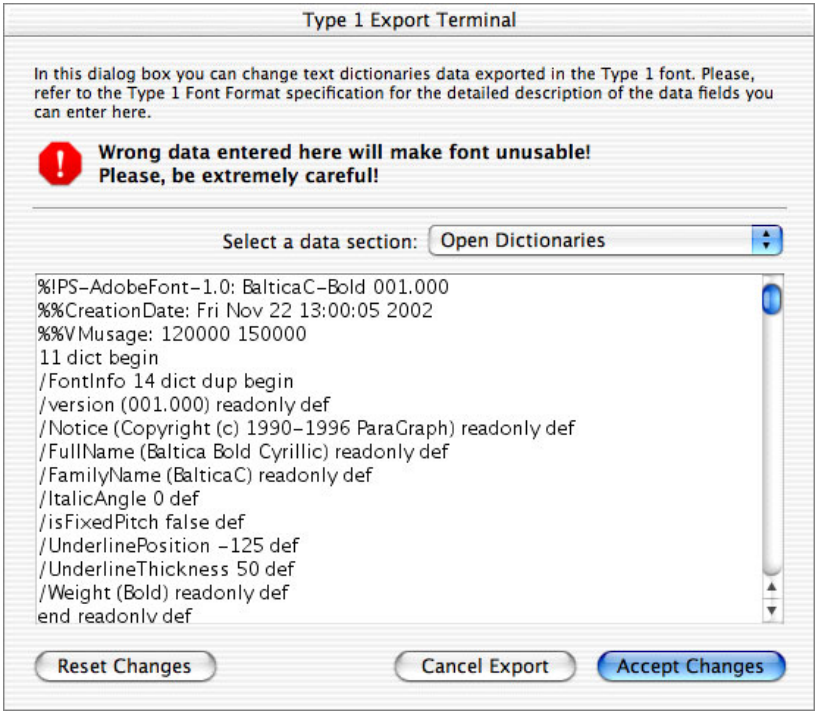
You can request several sizes of bitmap fonts in the Type 1 exporting options section:



Enter the point sizes of the bitmaps you want to export with the Mac Type 1 font, separating values by a space or comma.

Type 1 Export Terminal

If the **Open Type 1 Export Terminal** option is switched on and you save a font in Type 1 (Multiple Master or not) format, the following dialog box appears:


Type 1 Export Terminal
In this dialog box you can change text dictionaries data exported in the Type 1 font. Please, refer to the Type 1 Font Format specification for the detailed description of the data fields you can enter here.
Wrong data entered here will make font unusable!
Please, be extremely careful!
Select a data section:
Open Dictionaries
%!PS-AdobeFont-1.0: BalticaC-Bold 001.000
%%CreationDate: Fri Nov 22 13:00:05 2002
%%VMusage: 120000 150000
11 dict begin
/FontInfo 14 dict dup begin
/version (001.000) readonly def
/Notice (Copyright (c) 1990-1996 ParaGraph) readonly def
/FullName (Baltica Bold Cyrillic) readonly def
/FamilyName (BalticaC) readonly def
/ItalicAngle 0 def
/isFixedPitch false def
/UnderlinePosition -125 def
/UnderlineThickness 50 def
/Weight (Bold) readonly def
end readonly def
Reset Changes
Cancel Export
Accept Changes

As you can see, this dialog box previews and allows you to edit text data contained in the “open” and “private” sections of the exporting Type 1 font file. Use the **Select a data section** control to choose the part of the Type 1 font file that you want to edit and modify the text in the edit field below.

Note that there is no external control of changes you make in the export terminal. Use it only if you really know what you are doing or you may create a font which will not only be unusable, but can even crash your operating system.

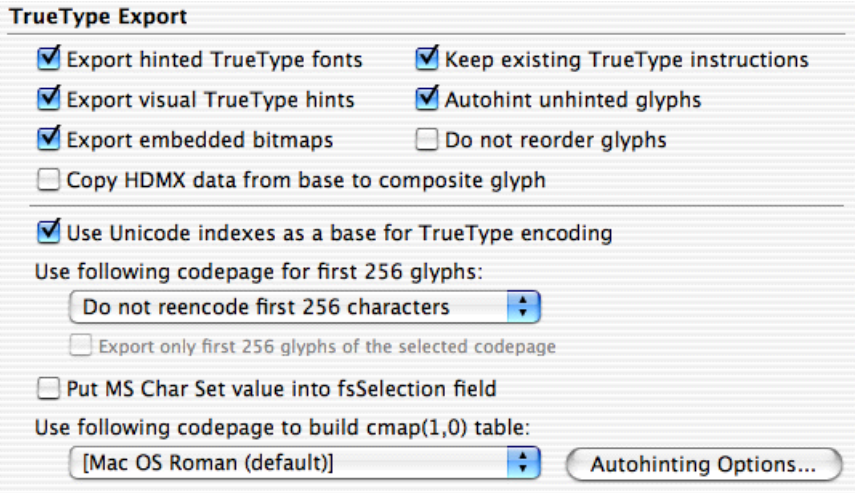
Refer to the Type 1 font format specification

http://partners.adobe.com/asn/developer/pdfs/tn/T1_SPEC.PDF

for information about the contents of the Type 1 font file.

TrueType Export Options

TrueType export options are concentrated in the **TrueType** page of the **Options** dialog box:


TrueType Export
Export hinted TrueType fonts
Keep existing TrueType instructions
Export visual TrueType hints
Autohint unhinted glyphs
Export embedded bitmaps
Do not reorder glyphs
Copy HDMX data from base to composite glyph
Use Unicode indexes as a base for TrueType encoding
Use following codepage for first 256 glyphs:
Do not reencode first 256 characters
Export only first 256 glyphs of the selected codepage
Put MS Char Set value into fsSelection field
Use following codepage to build cmap(1,0) table:
[Mac OS Roman (default)]
Autohinting Options...

The TrueType format is more complex and more flexible than Type 1, so you have many more options to customize.

There are two groups of TrueType export options: **glyph export** (including control over instructions) and **Unicode export**.

Glyph Options

Glyph export options control how FontLab exports glyphs and the first four options are related to the export of TrueType instructions:

Export hinted TrueType fonts	FontLab will export TrueType instructions of any type (original, manually edited or automatically generated) only if this option is on. To create a completely unhinted TrueType font switch off this option.
Keep existing TrueType instructions	If this option is on and the original TrueType instructions were saved when the font was imported, then FontLab will try to restore the original instructions where possible. If you want to drop all the original TrueType instructions, switch this option off.
Export visual TrueType hints	FontLab will convert manually set visual TrueType instructions only if this option is on.
Autohint unhinted glyphs	If this option is on, FontLab will try to automatically generate TrueType instructions for all unhinted glyphs.

How FontLab Autohints TrueType Fonts on Export

When autohinting is allowed and FontLab finds a glyph that has no TrueType hints of any kind (original hints imported from the TrueType font file or manually edited visual hints) it begins to make TrueType hints automatically. If this glyph has Type 1 hinting information, then FontLab converts this information to visual TrueType instructions and converts the instructions to the TrueType hinting code. If Type 1 hints are not present then FontLab automatically generates Type 1 hints as the first step, then converts the Type 1 hints into TrueType visual instructions and converts the visual instructions into TrueType native instructions.

The next three options control glyph ordering, export of embedded bitmaps and building an HDMX table:

Export embedded bitmaps	If this option is on, FontLab will export embedded bitmaps if they are present. If you don't want to save bitmaps in the font, switch this option off.
Do not reorder glyphs	If this option is off, FontLab will try to reorder glyphs to match the Mac cmap encoding table (this is a requirement of the specification). Activate this option to prevent glyphs from reordering.
Copy HDMX data from base to composite glyph	If this option is active, FontLab will copy HDMX data (device and point-size specific font metrics) from the base component of the composite glyph to the composite glyph. Of course, this will happen only if their metrics are the same.

Unicode Options

Most TrueType fonts use Unicode indexes for the glyph encoding information. Therefore if you want to make a TT font that will always work your Unicode glyph mapping tables must be perfectly correct.

Here is a description of the Unicode export options:

Use Unicode indexes as a base for TrueType encoding	If this option is on then FontLab will use the Unicode indexes that are set for the font's glyphs as the source of Unicode encoding information. See below.
Use the following codepage for the first 256 glyphs	FontLab can optionally reencode the first 256 glyphs of the font, letting you create a "single-codepage" font for some codepages. This is very similar to using the **Glyph/Glyph Names/Reencode glyphs** command before export. The popup menu contains several choices. The first option, "Do not reencode the first 256 characters" means that reencoding is off. All other options in the list let you select a remapping table to perform reencoding.
Export only the first 256 glyphs of the selected codepage	This option is enabled only if you choose one of the remapping codepages in the popup menu above. When this option is on FontLab will export only glyphs whose codes are in the 0-255 range.
Put MS Char Set value into fsSelection field	This is a "hack" used in older operating systems to correctly handle non-Latin fonts. It is not required and not recommended, but if you find that you need to use this hack to get your font to work in an old OS, just switch on this option and re-export the font.
Use the following codepage to build a cmap (1.0) table	Encoding information is stored in TrueType and OpenType fonts in a "cmap" table. Every Windows TrueType font contains at least two of these tables. One is the Unicode table and it "assigns" Unicode information to Glyphs. The other is a single-byte table that is used by older versions of the Mac OS and by some non-Unicode-compatible Windows programs. Use the control to select the codepage that will be used to build this table. There are two special options: Mac OS Roman (which is the codepage used on the Mac and the default choice) or Current codepage in the Font Window, which means that FontLab will use the encoding (yellow zone) currently selected in the Font window.

There are two possible methods of making TrueType Unicode mapping tables. *Unicode* mode (This works when **Use Unicode indexes as the basis for TrueType encoding** is on) and *Names* mode. In Unicode mode FontLab uses the Unicode indexes that you assigned to the glyphs while moving glyphs in the Unicode mode of the Font window or by entering Unicode indexes in the **Rename Glyph** dialog box. In the Names mode FontLab will encode all glyphs in the encoding currently selected in the Font window ("yellow" glyphs) and put them in the first 256 range of the TrueType Unicode table. FontLab will put all other glyphs into the unencoded zone of the Unicode table (starting from the code E000h).

The Unicode TrueType exporting mode is usually used to export fonts that were imported from TrueType font files and already have Unicode information. The Names mode is useful when you convert a Type 1 font to the TrueType format and do not want to worry about assigning Unicode indexes.

The general rule is simple: if you work with the Font window in Names mode use the Names TrueType exporting mode. If you work in Unicode mode use the Unicode exporting mode. Actually, FontLab will show a warning message if you try to export a font in Unicode mode while you are in the Names mode of the Font window and vice versa.

What is the "First 256 Glyphs Range"?

When we talk about the "first 256 glyphs" we do not mean those with Unicode indexes in the 0x0000 – 0x00FF range. We mean those that system will see as having codes 0x00-oxFF when the default codepage is selected. In most cases the default codepage is the Mac OS Roman codepage.

So even if you export a TrueType font in the Names mode the Unicode indexes for the encoded glyph will be selected from the Mac OS Roman codepage.

We recommend opening a TrueType font, switching to the Codepages mode of the Font window and selecting the Mac OS Roman codepage to better understand what the "standard" Macintosh glyphs are.

Windows Symbol Encoding

If you want to have full control over glyph codes and you want to occupy all 256 glyph cells of the first-byte range, you should create your font using the Windows Symbol codepage. The Symbol codepage begins from Unicode index F000h and continues to F0FFh. All glyphs in this range are mapped to the 0000h-000FFh range. So the glyph with code F041h will be seen in Windows as 'A' and so on.

To make a font using the Symbol codepage you have two options:

1. Switch the Font window to the Codepages mode; select the Windows Symbol codepage; and place all glyphs in the codepage. Export the font in TrueType format in Unicode mode.
2. Switch the Font window to the Names mode; select any encoding vector; place glyphs as you want in the "yellow" area; and export the font in Names mode (**Use Unicode indexes as the base for TrueType encoding** should be off).

In either case select **Symbol Microsoft Glyph Set** in the Codepages page of the **Font Info** dialog box. Without this the font will not be exported as Symbol and you will not have access to the glyphs at all.

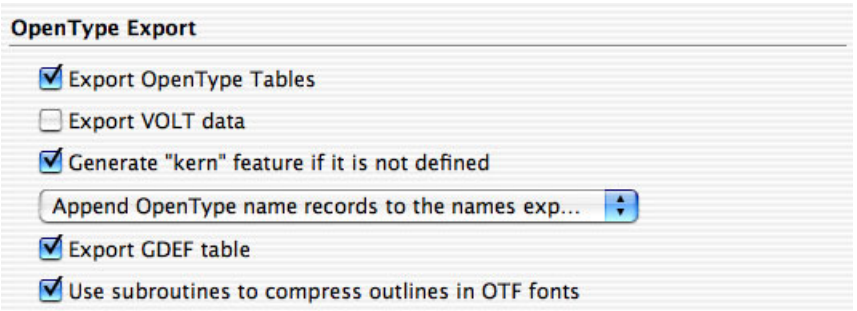
The Problem of Glyph 183, "middledot"

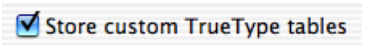
The glyph cell with decimal code 183 has different behaviors in Windows 95 and Windows NT. In Windows 95 it is mapped to the Unicode index 22C5h, but in Windows NT it is mapped to 00B7h. So, when FontLab exports your font in the Names mode, it automatically assigns both Unicode indexes to this glyph to make the font compatible with both operating systems.

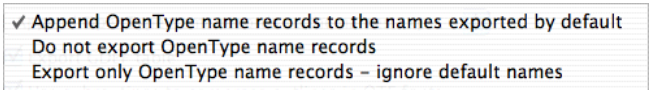
OpenType Export Options

As we said before, in structure there is not much difference between TrueType and OpenType fonts. The key difference is that OpenType fonts contain some additional tables that allow advanced typographic and lexical features. The additional data is stored in 3 tables: GDEF, GSUB and GPOS. We will discuss these tables in full detail in the "OpenType Fonts" chapter, so all you need to know here is that if you want to have OpenType features in your font, you have to let FontLab export these tables.

OpenType export features are stored in the OpenType page of the **Application/Preferences** dialog box:


OpenType Export
Export OpenType Tables
Export VOLT data
Generate "kern" feature if it is not defined
Append OpenType name records to the names exp...
Export GDEF table
Use subroutines to compress outlines in OTF fonts

Export OpenType Tables	Keep this option on to export OpenType tables. If it is off, you will get a plain TrueType font without any OT *features*.
Export VOLT data	VOLT is a Microsoft tool to define OpenType features. FontLab can try to export some of the features in VOLT format so you can open the font in VOLT and continue working on OpenType. Please note that you also need to have this option activated in the Import section of the TrueType page: 
Generate "kern" feature if it is not defined	The typical way to store kerning data in a plain TrueType font is to create a "kern" table that enumerates all the kerning pairs in the font. OpenType fonts define kerning differently, using the feature named "kern". Switch the option on to have FontLab automatically generate the "kern" feature from the kerning pairs table that you created in the Metrics window.

Name table export options	In FontLab you can define a basic set of font names that will be used if the font is previewed in an English environment. In addition you can define language-specific names that appear if the font is used in a system in which the environment is not English. The popup menu has the following options:  Choose one of these options to determine which names will be included in the font.
Export GDEF table	The GDEF table defines glyph types and the position of the caret inside ligature glyphs. FontLab provides visual tools to edit this information, so if you want it to appear in the exported font, leave this option switched on.

Recommended Export Options

We recommended selecting different export options in different situations. In addition to the export options we also suggest particular import options that will produce the best results in specific situations:

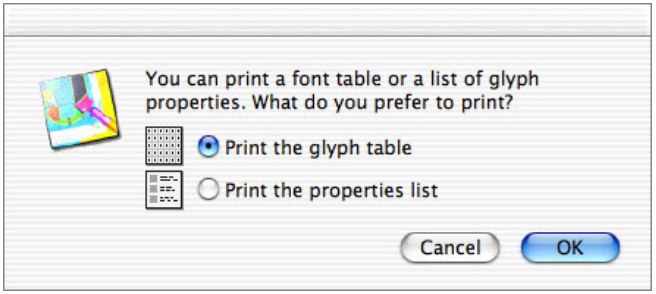
Source	Destination	Import options	Export options
TrueType	TrueType	Keep instructions, do not convert outlines, do not decompose, do not autohint	All hinting options — on, Unicode export, do not reencode
TrueType	Type 1	Keep instructions, convert outlines, do autohint, do not decompose,	Write PFM, AFM and INF files, Select encoding automatically. Before export, switch the Font window to the Names mode and select the desired encoding vector
Type 1	Type 1	Do not decompose, Generate Unicode	Write PFM, AFM and INF files, Select encoding automatically
Type 1	TrueType	Do not decompose, Generate Unicode	All hinting options — on, Names encoding export mode, do not reencode first 256 glyphs

Of course you can choose other options, but when you just want to convert a font from one format to another these recommended combinations of import and export options will usually give you fonts that will work fine in most environments.

Printing Fonts

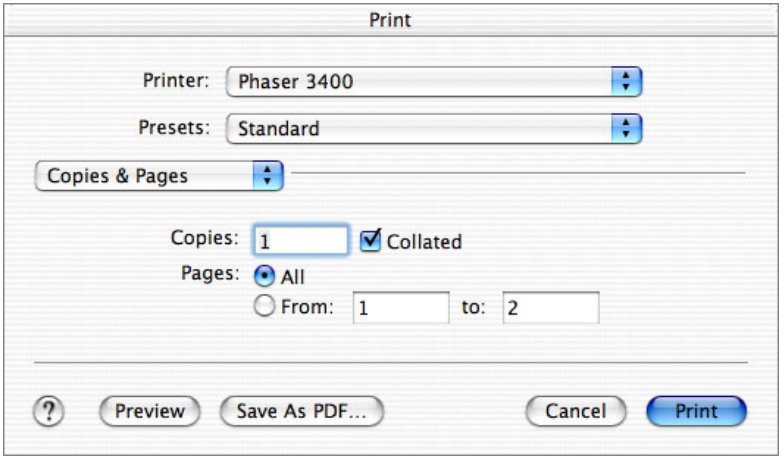
To print the font table select the **Print** command in the **File** menu.

You will see a dialog box that asks you to select one of the printing modes:


You can print a font table or a list of glyph properties. What do you prefer to print?
Print the glyph table
Print the properties list
Cancel
OK

Select “Print the glyph table” to print a font chart containing samples of all font glyphs or “Print the properties list” to print short descriptions of all glyphs currently selected in the Font Window.

After you make the choice you will see the standard **Print** dialog box that will ask you to choose a printer and modify the printer settings:

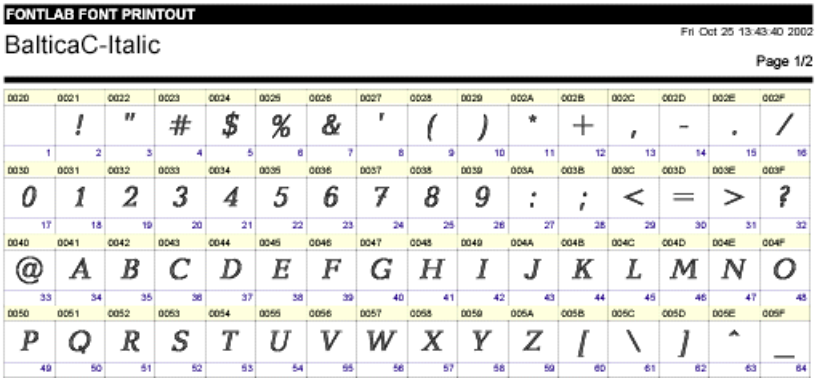


In this dialog you can choose the range of pages you want to print.

When you press the **Print** button, FontLab will print a font table containing samples for all font glyphs, their names, and codes according to the current encoding and glyph cell caption settings.

If you print a Multiple Master font the glyphs will appear according to the current font's WeightVector.

Printout of the font chart:


FONTLAB FONT PRINTOUT

Fri Oct 25 13:43:40 2002

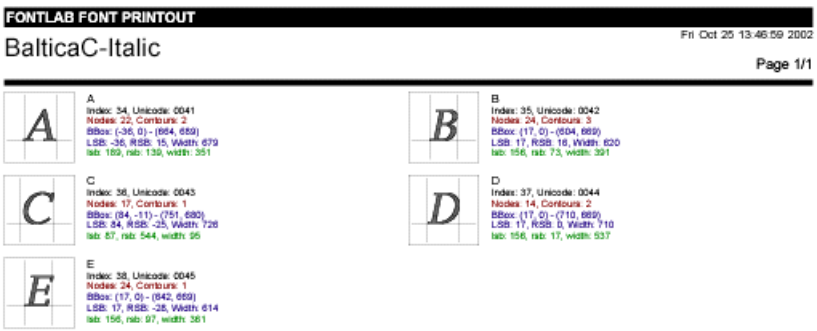
BalticaC-Italic

Page 1/2

0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	002A	002B	002C	002D	002E	002F
	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0030	0031	0032	0033	0034	0035	0036	0037	0038	0039	003A	003B	003C	003D	003E	003F
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0040	0041	0042	0043	0044	0045	0046	0047	0048	0049	004A	004B	004C	004D	004E	004F
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	005A	005B	005C	005D	005E	005F
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

Note that some information is colored; so printing on a color printer will have some benefits.

Printout of the glyph properties list:


FONTLAB FONT PRINTOUT

Fri Oct 25 13:46:59 2002

BalticaC-Italic

Page 1/1

A
Index: 34, Unicode: 0041
Nodes: 22, Contours: 2
BBox: (-36, 0) - (664, 669)
LSB: -36, RSB: 15, Width: 679
lsb: 169, rsb: 139, width: 351

B
Index: 35, Unicode: 0042
Nodes: 24, Contours: 3
BBox: (17, 0) - (604, 669)
LSB: 17, RSB: 16, Width: 620
lsb: 156, rsb: 73, width: 391

C
Index: 36, Unicode: 0043
Nodes: 17, Contours: 1
BBox: (84, -11) - (751, 680)
LSB: 84, RSB: -25, Width: 726
lsb: 87, rsb: 544, width: 95

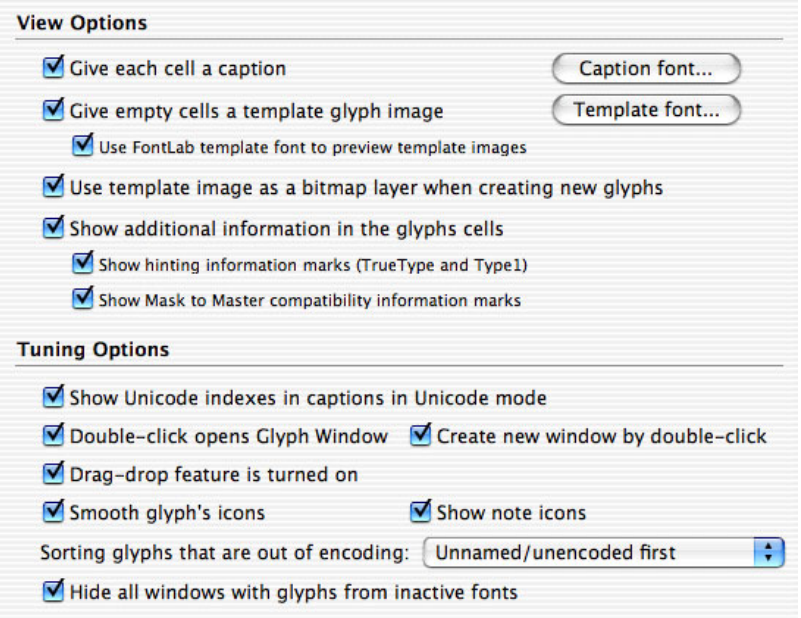
D
Index: 37, Unicode: 0044
Nodes: 14, Contours: 2
BBox: (17, 0) - (710, 669)
LSB: 17, RSB: 0, Width: 710
lsb: 156, rsb: 17, width: 537

E
Index: 38, Unicode: 0045
Nodes: 24, Contours: 1
BBox: (17, 0) - (642, 669)
LSB: 17, RSB: -28, Width: 614
lsb: 156, rsb: 97, width: 381

As you can see, this is the same information that is displayed in the Glyph properties panel.

Font Window Options

To change Font window options, select the Font Window page in the **Application/Preferences** dialog box:


View Options
Give each cell a caption
Caption font...
Give empty cells a template glyph image
Template font...
Use FontLab template font to preview template images
Use template image as a bitmap layer when creating new glyphs
Show additional information in the glyphs cells
Show hinting information marks (TrueType and Type1)
Show Mask to Master compatibility information marks
Tuning Options
Show Unicode indexes in captions in Unicode mode
Double-click opens Glyph Window
Create new window by double-click
Drag-drop feature is turned on
Smooth glyph's icons
Show note icons
Sorting glyphs that are out of encoding:
Unnamed/unencoded first
Hide all windows with glyphs from inactive fonts

Controls on this page are separated into two groups: View options and Tuning options.

View Options

Caption and Template Options

Use the check boxes to switch on or off the cell captions and the template in the empty cells. Use the **Choose font** buttons to select the fonts that appear in the caption and as templates in the empty cells.

If you switch on the option **Use FontLab template font to preview template images** FontLab will use its own bitmap font to show templates. The FontLab template font contains several thousand glyphs that cover almost all Unicode ranges except CJK (Chinese, Japanese and Korean) glyphs. We recommend using the FontLab template font when developing a Unicode font that is intended to support multiple (non-Asian) languages and codepages.

Use template image as a background layer when creating new glyphs

With this option FontLab will put the bitmap character from the Font window template into the bitmap background layer of a newly created glyph. This serves as an excellent reference of the glyph shape, especially for complex scripts.

Show additional information in the glyphs cells

Allows FontLab to preview glyph property information in glyph cells. This information is very useful if you want to take a brief look at your font and see the state of the glyphs. The two following options give you more detailed control over this feature.

Tuning Options

Show Unicode indexes in captions in Unicode mode

When this option is on FontLab will automatically show Unicode indexes in cell captions when you switch the Font window to the Unicode mode. It will restore the selected caption mode when you return to the Names mode.

Double-click opens Glyph Window

Switch this option off if you don't want FontLab to open a glyph in the Glyph Window when you double-click on the glyph cell

Create new window by double-click

If this option is on, FontLab will create a new Glyph window when you double-click a cell in the Font window characters chart. When it is off double-clicking a cell will create a new glyph window if there is none, but if there already is a glyph window then the glyph in it will be replaced by the glyph from the cell that was double-clicked.

Drag-drop feature is turned on

Switch this option off to switch off drag-drop editing in the Font window.

Smooth glyphs' icons

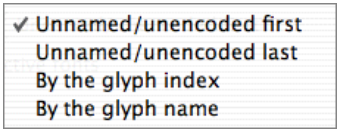
Activate this option to have glyph sample images in the Font window appear smooth. This increases visible resolution but requires more computing power. If you feel that the Font window is slow, try deactivating this feature.

Show note icons

If this option is on, Note icons will appear in the Font window.

Sorting glyphs that are out of the encoding

Use the options in this popup to choose how FontLab will sort glyphs that are not ("gray zone") in the current encoding ("yellow zone"):



Hide all windows with glyphs from inactive fonts

When this option is on FontLab hides all Glyph and Metrics windows with glyphs from any font whose Font window is inactive. This feature helps to avoid screen crowding with many unnecessary windows.

The Font Header

Perhaps the most important information you need to define for a font is its header or font info data. This information is mainly used to properly register the font in the operating system and in any program that uses it.

It is very important to carefully define all font parameters. Even the best-designed font is useless if it cannot be installed.

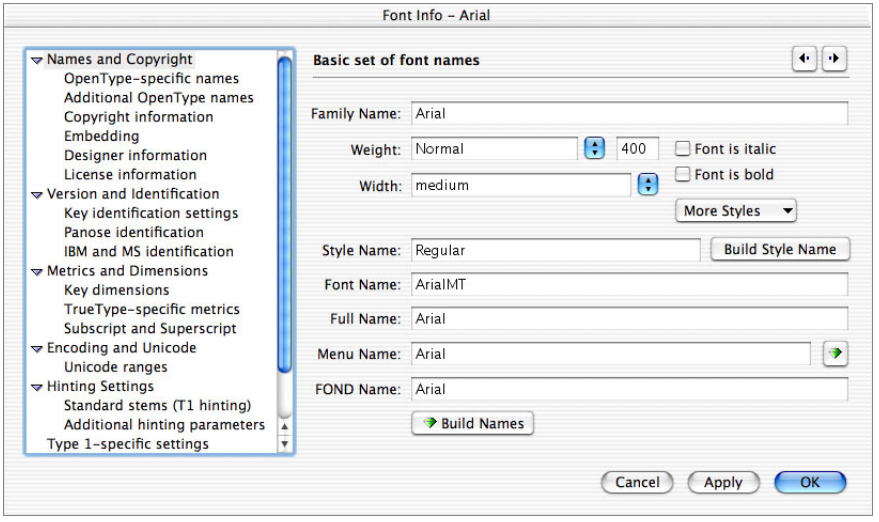
3

Font Info Dialog Box

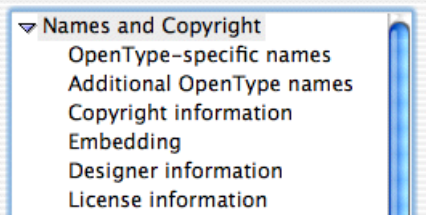
The control center where you define font parameters is called the Font Info Dialog box and is accessible from the **File** menu:

or with the button on any Font window:

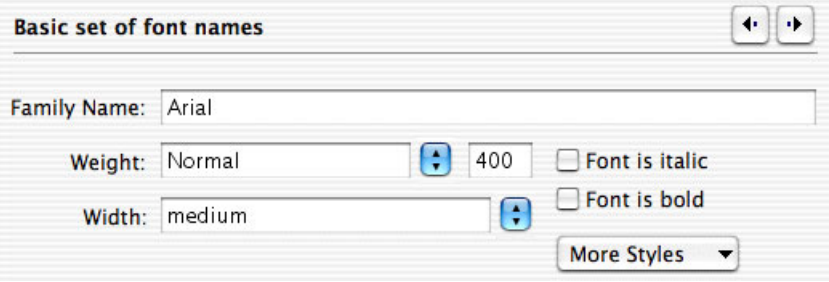
The Font Info dialog box consists of three parts:



At the left there is a page selection control where you can choose one of the sections in order to edit part of the Font Info information:



When you select one of the pages, it immediately appears to the right of the list:



Use the arrow buttons in the top-right area of the page to browse all available pages:

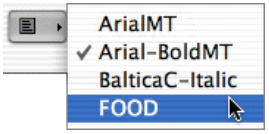
Alternately you may use the **Ctrl+Tab** and **Ctrl+Shift+Tab** key combinations to browse pages.

Command Bar

In the bottom area of the dialog box you find the command bar:

The Left part of the bar allows you to select a font for header information editing without closing the Font Info dialog box. Use the arrow buttons to browse open fonts:

or click this button to select a font from the list:



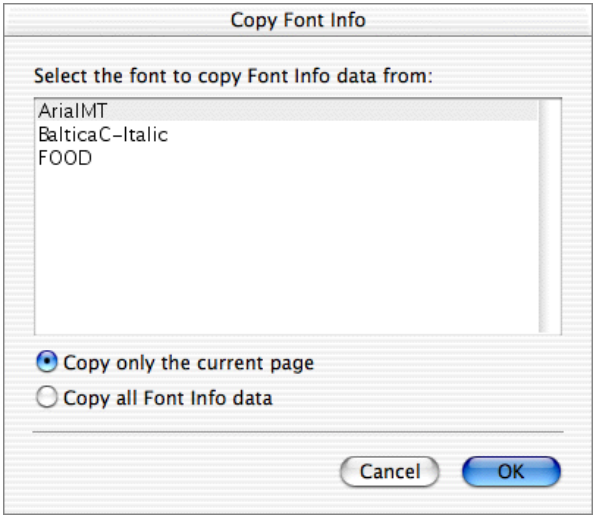
To the right you find four buttons:

Copy...	Allows you to copy font header information between open fonts.
Cancel	Cancels any changes and closes the dialog box.
Apply	Accepts changes but lets you continue, so you can see the results of your changes in the Font, Glyph or Metrics windows.
OK	Accepts changes you made to font info and closes the dialog box.

Copying Font Info

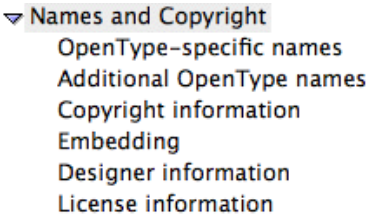
If you want to copy font info from one font to another, you can do it with the Font Info dialog box.

1. Open both fonts.
2. Activate the font **to** which you want to copy information.
3. Open the Font Info dialog box.
4. Click on the **Copy** button.
5. In the dialog box select the font from which you want to copy information:



Using the options below the list you can copy only the information from the current page or the whole font info. Make a selection and click **OK** to complete.

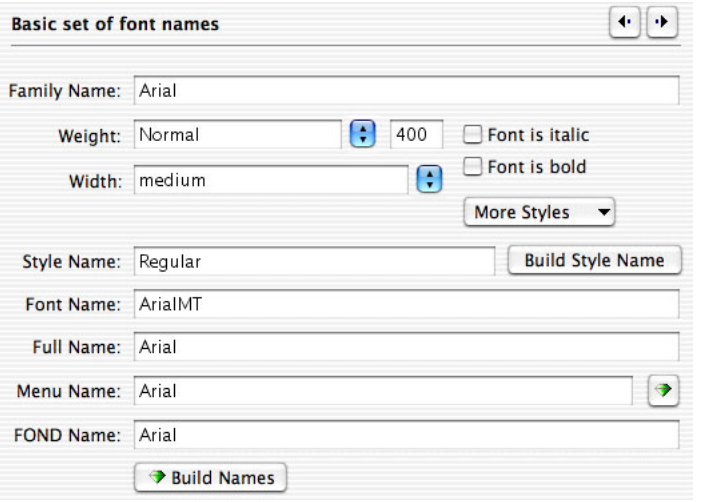
Font Names



The names section includes the most important font-registration information.

All programs use the information on this page to refer to a font. Be sure to enter all the values very carefully and use the automatic features where available.

Basic Identification and Names



Family Name	[name: 1] (this mark is the ID of the name in the TrueType and OpenType specification: http://microsoft.com/typography/otspec/name.htm) The name of the typeface to which the font belongs. All fonts that are from the same typeface must have the same **Family Name** field. The Family Name is used as the root of the Font Name so we recommend that you fill in this field first.
Weight	Weight of the font. You may enter a custom value in this field or select one of the predefined weight names in the popup. Values in this list are sorted by increased weight value. Choose **Normal** or leave this field empty if you do not care about the font's weight.
Weight Value	Numeric weight value of the font. This number defines the font weight and is used by the operating systems to organize fonts to font families. FontLab will fill it automatically when you select some Weight in the popup menu, but if you want you can customize it.
Width	The average width of the font's characters. Enter a custom value or select one of the predefined width values from the popup menu. Leave this field empty or select **Normal** width if you do not care about the font's width.
Font is Italic	[OS/2: fsSelection] Switch on this check box if you are creating an italic font.

Font is bold	[OS/2: fsSelection] The Font is defined as bold. Usually this checkbox is related to the Weight setting, but it is not required. For example, if you are making a family containing Light and Normal styles, you may need to mark the Normal style as Bold so you will not need to split these styles into two separate families.
More styles	Press this button to open a popup menu where you can select one of the additional font styles. This information is used only by TrueType fonts, but we recommend you always set it properly to simplify future font identification.
Style Name	[name: 2] Contains complete style information about the font. We recommend that you fill in the Weight, Width and Italic data, to automatically generate this field using the **Build Style Name** button and edit this field if necessary.
Build Style Name	Press this button to automatically generate the **Style Name** field. Style names are based on the Width, Weight and Italic information.
Font Name	[name: 6] PostScript name. This name will be used by a PostScript print driver to reference the font. Do not include spaces in this name.
Full Name	[name: 4] More detailed font name. It may include spaces as well as any other characters – this is the name that is exposed to users when the font is installed in Windows and sometimes in Mac OS.
Menu Name	The name used to access the font in applications. This name must not include style information (bold, italic or similar). The length of this field is limited to 31 characters for TrueType or single-master Type 1 fonts and to 7 characters for Multiple Master fonts. To ensure that the current Menu name is made properly, press the **Check** button.
FOND Name	This name is used by the Mac OS to organize fonts into font families. Windows does not use it. We recommend you fill in this name if you plan to use your font on Mac.
Build Names	Press this button to automatically generate the **Font Name, FOND Name** and **Full Name** fields. If you are creating a new font we recommend that you fill in the **Family Name** field, generate or manually fill in the **Style Name** field and press this button to create the Font, FOND and Full names. If necessary you can edit the names later.

How to Make a Font Family

If you have several styles of one font family (typeface), you should follow these simple rules:

Enter the name of the font family in the Family Name record. Enter the same name in the Menu Name record, but control the length of the Menu Name by pressing on the **Check** button. The Family Name and Menu name should be the same for all fonts of the family.

Select the correct styles for all fonts. Choose Normal or Regular weight for the normal style; check the Font italic option for the italic or oblique style; and use the Font is bold option to define a style as bold. It is recommended (but not required!) that you select the proper weight and width names. You can enter any name, but we highly recommend using only the standard names from the list.

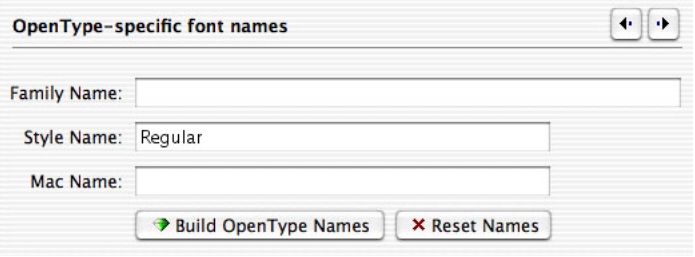
Fill in the **Style Name** field. We recommend using the **Build Style Name** button.

Fill in the **Font Name, FOND Name** and **Full Name** fields. The best way is to press the **Build Names** records button and let FontLab fill in all the fields automatically.

Note 1: Fonts in both Type 1 and TrueType formats may include only 4 styles in a family. These styles are: normal, bold, italic and bold-italic and their names are very standard.

If you have more than four styles in your typeface you must create several families. You may put all condensed styles into a Condensed or Narrow *sub-family* (like Arial Narrow, Arial Narrow Bold, Arial Narrow Italic), all black styles into a Black sub-family (Arial Black, Arial Black Italic) and all "normal" styles in the "Normal" sub-family (Arial, Arial Italic, Arial Bold and Arial Bold Italic).

OpenType-Specific Names



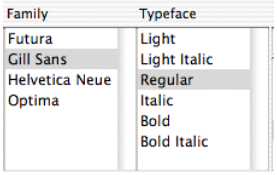
The OpenType format adds several new name records that are needed to use OpenType fonts on PC and Mac:

Family Name "Preferred Family Name" [name: 16] –This name is used to create a family containing more than 4 styles. You need to use the same Preferred Family Name in all fonts that you want to put into a "big" family and make the Preferred Style Name different for each of these fonts.

The Preferred Family name appears in the font menu as the "font name".

Please note that this information is used only by new applications that can handle OpenType fonts. Adobe InDesign or other new Adobe programs are good examples.

Style Name "Preferred Style Name" [name: 17]– Used to complement Preferred Family Name and defines a font style in a big font family. This is the name that appears in the submenu of the fonts list:

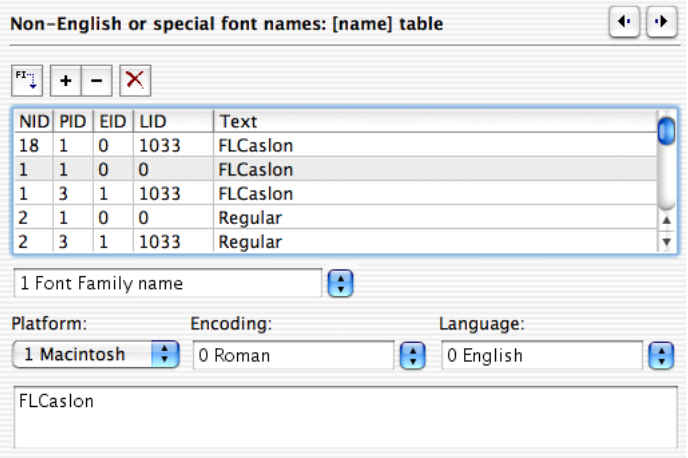


You need to include the Preferred Family Name and Preferred Style Name on this page only if they are different from the Family and Style names you defined on the Basic Names page.

Mac Name Macintosh compatible full name [name: 16] – If you want the name of the font to appear differently than the Full Name (defined on the Basic Names page), you can insert the Compatible Full Name.

➲ **Note:** You don't need to fill in the **Preferred Family name** and **Preferred Style Name** fields if they are the same as the Family Name and Style Name on the main naming page of Font Info. Enter names there only if the names are different.

Non-English and Special Names

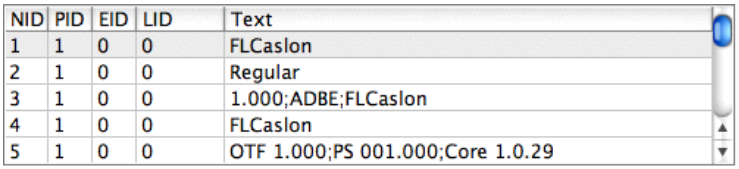


The TrueType and OpenType specifications let you put many names into the font. Most of them can be defined on specialised pages of the Font Info dialog box (Basic Names, OpenType names, Copyright, etc.) but these names are in English. If you need to have non-English names or names that are not covered by the FontLab FontInfo pages you may use this Non-English or special font names page.

Please refer to the OpenType specification to get full information about TrueType and OpenType names:

http://www.microsoft.com/typography/otspec/name.htm

Most of the page is covered by the names list:



NID	PID	EID	LID	Text
1	1	0	0	FLCaslon
2	1	0	0	Regular
3	1	0	0	1.000;ADBE;FLCaslon
4	1	0	0	FLCaslon
5	1	0	0	OTF 1.000;PS 001.000;Core 1.0.29

The Columns of the list are:

NID	Name ID – code of the name (this is what we put in the [name: x] comment on 'x' place).
PID	Platform ID – platform identified. Could be 0, 1, 2 or 3.
EID, LID	Encoding and Language IDs – see the OpenType name table specification.
Text	Content of the name table record. May include Unicode characters after the '\'.

Click on the caption of the column to sort name records by one of the values.

Below the list is a set of controls that let you define records.

Above the list there is a toolbar that you can use to modify the names table; add or remove name records; or import custom name records from the "English" data:

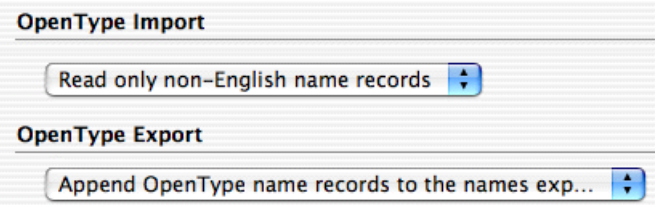
	Import names from the basic set of names defined in other pages of the Names section
	Add a name record
	Remove the currently selected name record
	Remove all custom name records

To add a new name record, click on the + button on a toolbar, select name record options using the controls below the list and type in the name record content. To enter non-ANSI characters use "\[unicode index]" or "\[code]" notation where [unicode index] is the Unicode index of the character if the name record is for the Unicode (0 or 1) platform and [code] is the character code in the Mac Roman codepage if the name record is for the Macintosh platform. For example you may enter "\0411" for the Cyrillic "beh".

Note that you can use the same notation in FontLab's Metrics window or Preview panel, so if you open a Unicode font you may test the name records there.

To remove a record select it in the list and click on the − button. Click on the ✕ button **to remove all name records**.

You can use the OpenType page of the Preferences dialog box to control the import and export of the additional name records:



Import options:

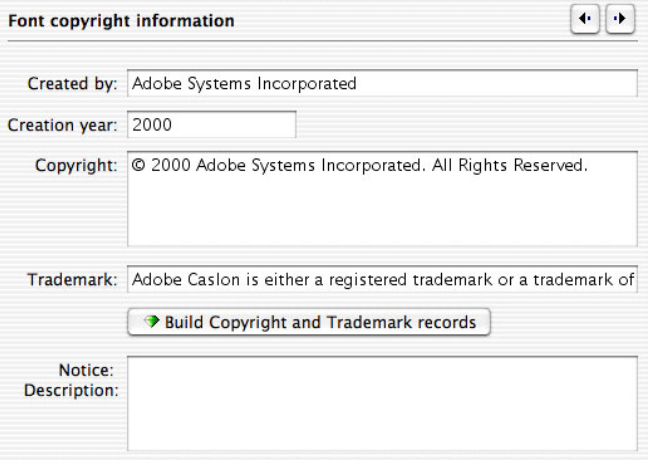
Read only non-English name records	This is the default choice – FontLab will read only those additional records that cannot be interpreted to "standard" name records, which have dedicated pages in the Font Info dialog box.
Do not read OpenType name records	With this selection any record that cannot be interpreted by the default algorithm is ignored.
Read all OpenType name records	All name records are imported as additional OpenType records and are placed in the page.

Export options:

Append OpenType name records to the names exported by default	This is the default choice: FontLab will export English names and then add only those additional OpenType names that aren't already covered.
Do not export OpenType name records	With this option FontLab will not export any additional name records.
Export only OpenType name records - ignore default names	When this option is selected, only additional name records are exported, all other names (from other pages in the Names section) are ignored.

➲ **Final note:** please, use this page carefully – FontLab doesn't verify information that you put into additional name records. Be sure that you read and understand the name table specification.

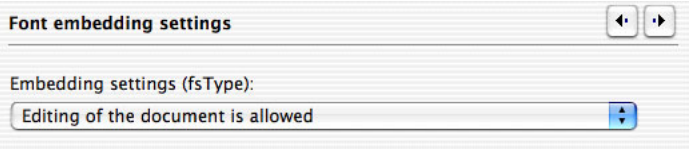
Copyright Information



On the copyright page you can enter information about the creators of the font. If you have created a new font you should enter your copyright notice here. If you have edited an existing font that was not your creation you must not remove the information contained on this page, or you may violate copyright laws.

Created by	[name: 8] Name of the company or person that created the font. If you are creating a new font enter your name or the name of your company here.
Creation year	Year when the font was created. This is used by FontLab to automatically fill in the **Copyright** field and is exported in TrueType fonts as the Creation year entry.
Copyright	[name: 0] Copyright message. Must include the © sign or the word "Copyright", the name of the company or person that owns the copyright and the copyright year. In Type 1 fonts this information is stored in the Notice entry and in TrueType fonts in the Copyright entry.
Trademark	[name: 7] Font trademark – used to save font's trademark notice.
Build Copyright and Trademark Records	Press this button to create the standard Copyright record based on the **Created By** and **Creation Year** fields.
Notice Description	[name: 10] Additional information that you want to include in Font Info. Exported in Type 1 fonts as the Copyright entry and in TrueType fonts as the Description entry.

Font Embedding



These settings control how the font may be embedded into documents. Embedding is a feature of the operating system and some applications that allow programs to include fonts into documents to guarantee that they will be reproduced correctly. However, this feature may cause problems with font piracy. It is not very hard to extract embedded fonts from a document, so the TrueType font format includes a special setting that can control font embedding.

There are four types of font embedding:

Everything is allowed	After the document is opened the font works as if it was installed in the system.
Embedding is not allowed	Embedding is not allowed for this font.
Only printing and previewing is allowed	The font may be embedded, but editing of the document it contains is not allowed.
Editing of the document is allowed	The font may be embedded and the document that contains the font may be viewed, printed and edited.

Copyright Note

We decided to allow modification of the embedding setting only because we are sure that the users of FontLab are professionals who respect others' rights to intellectual property. We assume that you will change the embedding setting **only in your own fonts**.

You are not allowed to change this setting in fonts that were created by somebody else. Even if according to the font license you can modify the font for your own use, you must not "increase" the embedding rights for a font. So if embedding is not allowed leave it as it is.

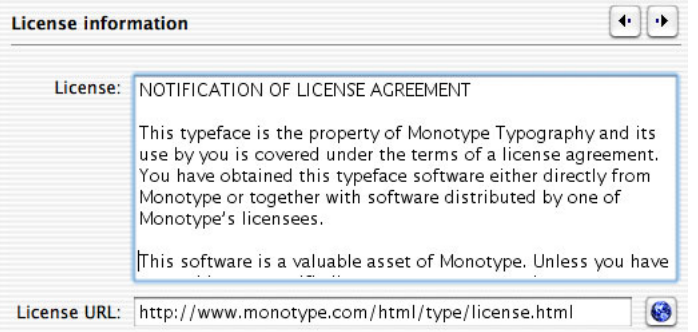
Designer Information

This page stores information about the font's designer. Do not modify this data if you open an existing font to modify for personal use.

Designer	[name: 9] Name of font designer
Designer URL	[name: 12] A new entry implemented only in TrueType format. It is the WWW link to the designer of the font.
Vendor URL	[name: 11] This TrueType-only entry shows the WWW link to the site of the font vendor.

Use the buttons to the right of the Designer URL and Vendor URL controls to open pages in your default Web browser window. This requires an Internet connection.

License Information

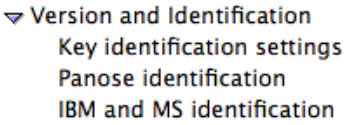

License information
License:
NOTIFICATION OF LICENSE AGREEMENT
This typeface is the property of Monotype Typography and its use by you is covered under the terms of a license agreement. You have obtained this typeface software either directly from Monotype or together with software distributed by one of Monotype's licensees.
This software is a valuable asset of Monotype. Unless you have
License URL:
http://www.monotype.com/html/type/license.html

License and License URL records are relatively new and have appeared only in OpenType specification version 1.3.

License	[name: 13] License description – contains information about how the font can be used
License URL	[name: 14] URL where additional license information can be found.

Use the button to the right of the License URL control to open the page in a Web browser window. This requires an Internet connection.

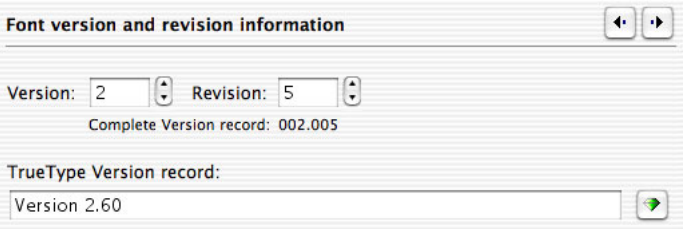
Font Identification



Sometimes the operating system or a DTP application needs to know what the font looks like. It may be necessary, for example, to properly substitute for a missing font with the closest lookalike.

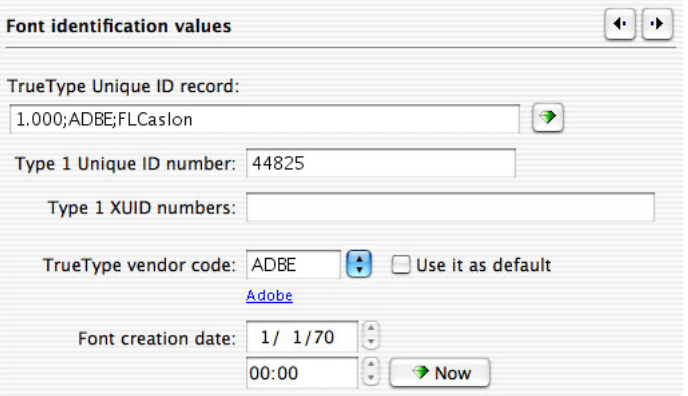
FontLab supports all the font-identification settings that are used in Type 1 or TrueType fonts.

Version Information



Version	Version of the font.
Revision	Revision of the font. Version and revision numbers are combined and build a complete version record that appears in Type 1 font headers.
TrueType Version Record	[name: 5] TrueType font version records have a different format. You may enter the TrueType version record here or just press the **Recalc** button at the right of the field to fill this record automatically. You must have the **Names** and **Copyright** pages filled in to use the automatic features on this page. Press the **Apply** button at the bottom of the dialog box to enter the new Font Info values into the font's header.

Basic Font Identification



TrueType Unique ID Record

This field is necessary to identify TrueType fonts. Usually it includes the creator's name, font family name and creation year. The format of this field is freeform, but we recommend that you use the button to fill this field automatically.

Type 1 Unique ID Record

An integer number identifying the font. Unique ID numbers must be registered with Adobe Systems. However, you may leave 0 in this field or enter a value from the users Unique ID zone (4000000 to 4999999). If you enter this value and plan to export Type 1 fonts, be sure not to have more than one font with the same Unique ID value because that may cause a problem with PostScript printers or Adobe Type Manager software.

Type 1 XUID Numbers

More advanced identification codes for Type 1 fonts. This number is used only in PostScript Level 2 printers. Please, refer to Adobe documentation for more information concerning the **XUID** field.

TrueType Vendor Code

An up-to-four letter length code that is assigned to most TrueType producers to identify their fonts. An *uppercase* vendor code must be registered with Microsoft or Apple. All registered Vendor codes known at the time of FontLab's release are placed in the popup menu. If you want to identify yourself without registering you may enter a *lowercase* four-letter vendor code.

Below the vendor selection list you can see the full name of the registered vendor. Click the name to open the vendor's page in a Web browser.

Use it as default	Check this option to use the current vendor code as the default in all new fonts. You may make your own code the default so you will not have to enter it every time.
Creation date and time	Use these controls to edit the font creation date and time. Press the **Now** button to set the current date and time.

Vendor.dat File

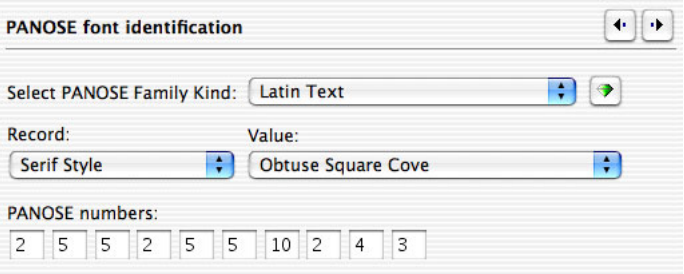
FontLab stores information about registered vendors in the vendor.dat file located in the FontLab/Data folder. This is a text file with a simple structure:

2REB 2Rebels
39BC Finley's Barcode Fonts
3ip Three Islands Press
918 RavenType

As you can see, it is just a vendor code followed by vendor name. A single space is used as a separator.

If you want to change the file or add a new entry, just open it in any text editor (such as SimpleText or TextEdit) and make changes.

PANOSE™ Identification



In the PANOSE identification system 10 numbers describe a font. Each number represents one identification category. The most important category (represented by the first number) defines the "kind" of font — is it a normal Roman font, a hand-written font, a decorative or symbol font. The meaning of all the other categories depends on this setting.

In all categories the first two values (0 and 1) mean Any and No Fit. Any means that the value of this category is not important. No Fit means that the category value for this font is not among the available values.

To set the PANOSE identification numbers, first select the font family kind.

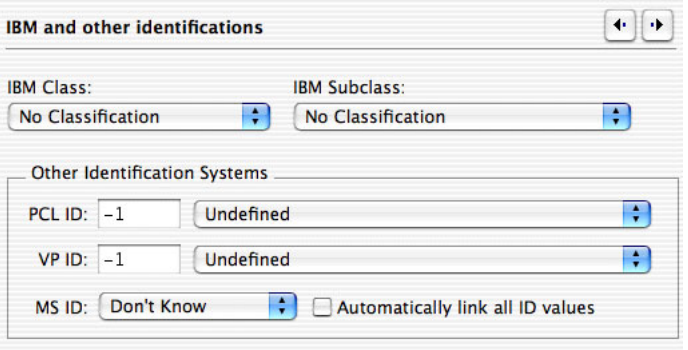
After that, select the category in the **Record** popup menu and the value in the **Value** popup menu.

Here is a table of the categories and possible values for the Latin Text family:

Serif Style	Cove, Obtuse Cove, Square Cove, Obtuse Square Cove, Square, Thin, Oval, Exaggerated, Triangle, Normal Sans, Obtuse Sans, Perpendicular Sans, Flared, Rounded
Weight	Very Light, Light, Thin, Book, Medium, Demi, Bold, Heavy, Black, Extra Black
Proportion	Old Style, Modern, Even Width, Extended, Condensed, Very Extended, Very Condensed, Monospaced
Contrast	None, Very Low, Low, Medium Low, Medium, Medium High, High, Very High
Stroke Variation	No Variation, Gradual/Diagonal, Gradual/Transitional, Gradual/Vertical, Gradual/Horizontal, Rapid/Vertical, Rapid/Horizontal, Instant/Vertical, Instant/Horizontal
Arm Style	Straight Arms/Horizontal, Straight Arms/Wedge, Straight Arms/Vertical, Straight Arms/Single Serif, Straight Arms/Double Serif, Non-Straight Arms/Horizontal, Non-Straight Arms/Wedge, Non-Straight Arms/Vertical, Non-Straight Arms/Single Serif, Non-Straight Arms/Double Serif
Letterform	Normal/Contact, Normal/Weighted, Normal/Boxed, Normal/Flattened, Normal/Rounded, Normal/Off Center, Normal/Square, Oblique/Contact, Oblique/Weighted, Oblique/Boxed, Oblique/Flattened, Oblique/Rounded, Oblique/Off Center, Oblique/Square
Midline	Standard/Trimmed, Standard/Pointed, Standard/Serifed, High/Trimmed, High/Pointed, High/Serifed, Constant/Trimmed, Constant/Pointed, Constant/Serifed, Low/Trimmed, Low/Pointed, Low/Serifed
X-height	Constant/Small, Constant/Standard, Constant/Large, Ducking/Small, Ducking/Standard, Ducking/Large

A detailed description of the PANOSE categories and values (the so-called Gray Book) may be found on the PANOSE web site at:

Other Identification Systems



IBM Identification

The IBM Identification system uses a different approach from the one used in the PANOSE system. This system is based on the font's appearance. In the IBM system two numbers that designate the font category and sub-category identify each font. A font's design is compared to one of the "standard" and well-known designs.

To set IBM Identification information select a Category that matches your font from the IBM Class list. Then select a sub-category in the IBM Subclass list.

A detailed description of the IBM Font Identification system may be found in the TrueType font format specification on the Microsoft Typography Web site at:

http://www.microsoft.com/typography/tt/tt.htm

PCL and Ventura Publisher Identification

These identification systems are relatively old ones and are rarely used in modern DTP applications. However, to keep compatibility with all possible usage of your font we recommend setting these options. They are used only with Type 1 fonts and are exported only in the INF file.

Both systems are based on font appearance so they are close to the IBM identification system. And both systems are one-level, so you just have to select a well-known font that is close in appearance to your font. To simplify this when the **Automatically link all ID values** check box is active and you select one of the values FontLab will select the a similar value in the other combo box. For example, if you select Helvetica in the **PCL ID** popup menu, Swiss will be selected in the **VP ID** popup menu.

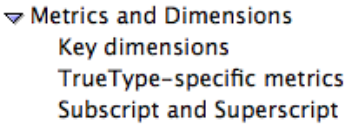
If you want you may enter PCL ID and VP ID indexes manually using the edit fields at the left of the popup menus. Please refer to the technical specifications for detailed descriptions of these identification systems.

Microsoft Identification

This is the simplest identification system. Just select one of the common font categories (Roman, Swiss, Modern, Script and Decorative) and you are done. This is the only identification system that is implemented in the PFM files that are used by Windows to work with Type 1 fonts, so we strongly recommend selecting the proper value in this list box.

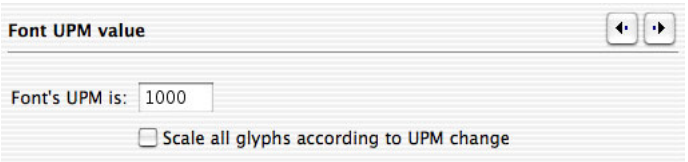
➲ **Note:** If you are building a Type 1 font for Windows, take this field very seriously. Do not leave “Decorative” in this option if the font is not decorative – this will affect the font spacing.

Metrics and Dimensions



This page is used to set font dimensions that are used mostly to properly align text lines.

Font UPM Value



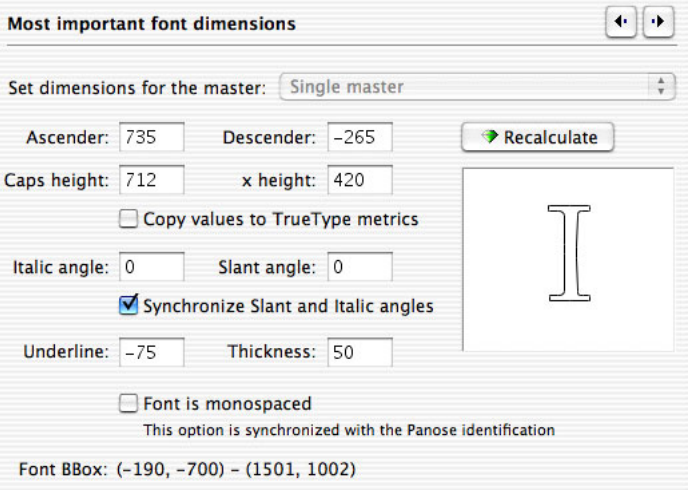
The most important field on this page is the **Font UPM**. Let's explain what UPM is and why it is so important.

The UPM (Units Per eM) is the basis of all font dimensions. The UPM is the number of font units that defines the font height and the coordinate grid on which the glyphs are drawn.

The bigger the UPM is the more coordinate space you have, so you can set more precise positions of points. For technical reasons in FontLab the UPM is limited to 10000 units, but we strongly recommend you work with one of the standard UPMs. In Type 1 fonts the standard UPM is 1000 units and in TrueType fonts the UPM may be set to any value, but the recommended value is 2048 units.

If you change the UPM value in the Dimensions page of the Font Info dialog box this doesn't necessarily mean that the size of the characters will change. For example, if you change the UPM from 1000 to 2000 all the glyphs will now be half as big as they were before – because they are still dimensioned at 1000 UPM. You have to scale all the characters to fit them in the new UPM setting manually or you can switch on the **Scale all characters according to UPM change** check box and all the font data will be scaled automatically.

Basic Font Dimensions



The page has several editing fields with numbers and a sample window where an appropriate character is displayed to help set correct values.

Other fields on this page mean:

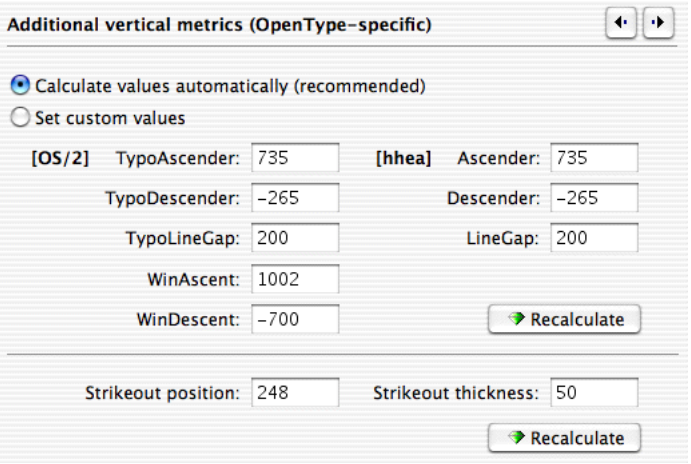
Ascender	Position of the font's ascender line. Usually this is the height of the lowercase 'b' character.
Descender	Position of the font's descender line. Usually this is the position of the bottom line of the 'p' character.
Caps height	Height of the font's uppercase characters. Usually the height of the 'H' character.
x height	Height of the lowercase characters. Usually the height of the 'x' character.
Italic angle	Actual italic or oblique angle for the font. The italic angle is measured in the counterclockwise direction, so the default value is -12°.
Slant angle	Type 1 fonts can be artificially slanted to get an "oblique" appearance while keeping the actual outlines upright. Enter a slant angle value (in degrees) here and check the result in the Preview panel.
Underline	This is the position of the middle of the underline line in your font.
Thickness	This is the thickness of the underline line.

If you press the **Recalculate dimensions** button, FontLab will automatically recalculate all the dimension values.

Ascender, Descender and Type 1 fonts

If you are making a Type 1 font you should set the Ascender and Descender values very carefully. In Type 1 fonts these values are used very directly to calculate interline spacing. It is usually necessary to set the Ascender value higher than actual height of the "ascender" 'b' character, to have some additional space between lines.

Advanced Vertical Metrics



In TrueType font files vertical metrics can be stored in the OS/2 and hhea tables. Different programs and operating systems use vertical metrics from these tables. Windows usually uses data stored in the OS/2 table while the Mac OS uses only data located in the hhea table.

It is important to correctly define all vertical metrics if you want your font to align properly. In most cases FontLab can calculate vertical metrics according to the system recommendations, but in some cases you may want to customize these values.

We recommend you generally leave these values untouched in an existing OpenType font. Of course, if you perform heavy modification of the font you will need to update the advanced vertical metrics.

If you want FontLab to automatically calculate all vertical metrics, select the option **Calculate values automatically**.

If you want to customize values, select the **Set Custom values** option and edit data in the fields below. Note that if you choose Set Custom values but leave all data unchanged, FontLab will restore the original vertical metrics data from the imported font and the new updated font will align exactly as the original one.

Here is a description of each value:

Typo Ascender	This is the typographically correct ascender value. It is the topmost line of lowercase characters, usually, the topmost line of the 'b' character.
Typo Descender	The same as Typo Ascender, but for the lowest line. Usually it is equal to the bottom line of the character 'p'.
Typo Line Gap	"Typographically" correct line gap value (distance between bottom line of the upper line of text and top line of the lower line of text).
WinAscent	[OS/2] This value defines the topmost line of all important characters in the font. "Important" characters are all non-exceptional characters. For example, if most of the characters have the topmost position at 900 font units and one, not often used character, has it at 1300 font units, it's a good idea to set WinAscent at 900 units. Note that in most cases portions of the characters that are above the WinAscent value will not appear on the screen or print on some printers. Please note that WinAscent is NOT a typography ascender, usually measured as the topmost line of lowercase characters. It is mostly a technical parameter used by the rasterizer to allocate vertical space to render characters.
WinDescent	[OS/2] The same as WinAscent, but for the lowest line of all "normal" characters.
Ascender	[hhea] This value is used by the Mac OS in about the same situation as Windows uses the WinAscent value from the OS/2 table – to define the topmost position of all important glyphs.
Descender	[hhea] In short: the Macintosh version of the Windows WinDescent parameter. If there are any pixels below this line the glyph will be squashed in the vertical direction to match metrics defined by the Ascender and Descender parameters.
Line Gap	[hhea] This value is used by the Mac OS to compensate Ascender and Descender values and calculate the correct distance between baselines of the text. Refer to the formulas below to see how baseline-to-baseline distance is calculated.

Baseline-to-baseline distance calculation

Windows:

Windows Metric	OpenType Metric
ascent	WinAscent
descent	WinDescent
internal leading	WinAscent + WinDescent – UPM
external leading	MAX(0, LineGap - ((WinAscent + WinDescent) - (Ascender - Descender)))

BTBD = ascent + descent + external leading

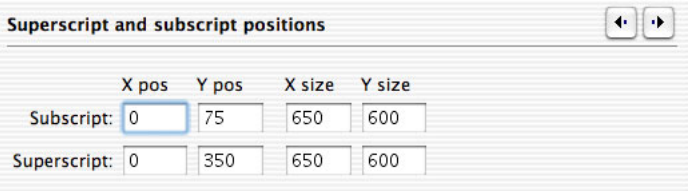
It should be clear that the "external leading" can never be less than zero. Pixels above the ascent or below the descent will be clipped from the character; this is true for all output devices.

Macintosh:

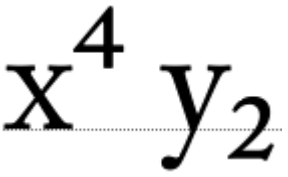
Macintosh Metric	OpenType Metric
ascender	Ascender
descender	Descender
leading	LineGap

BTBD = ascender + descender + leading

Superscript and Subscript

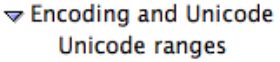


TrueType and OpenType font format allows you to specify position and size of the superscript and subscript characters:



Subscript	Position and size of the subscript characters in font units.
Superscript	Position and size of the superscript characters in font units.

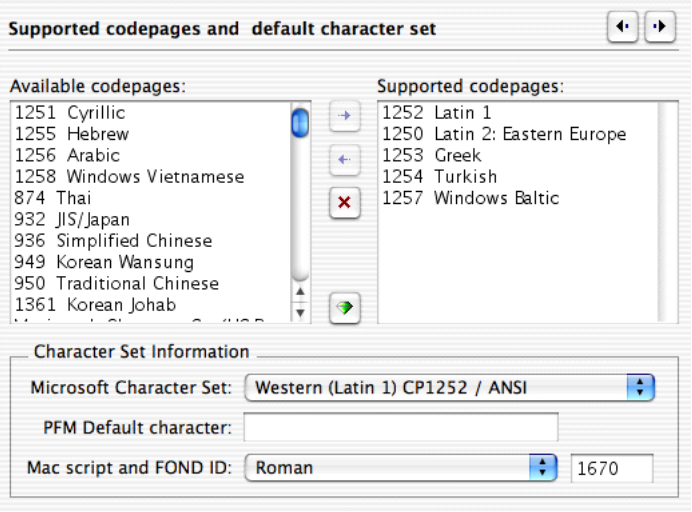
Encoding and Unicode



As we mentioned earlier, fonts may have very many characters and support a lot of different languages. To tell the operating system what codepages the current font can support, you set the codepages information.

TrueType and Type 1 fonts use different methods to identify what codepages a font supports. In TrueType fonts you can identify all the supported codepages by setting bits in a special field of the font header. In Type 1 fonts you select only one codepage (actually, encoding vector) and it must be compatible with the actual font encoding.

Supported Codepages



The operating system needs to know which codepages a TrueType font can support. To set this information you select all the codepages that this font can "cover" from the list of standard codepages that are available to the operating system.

To select the supported codepages automatically press the **Auto** button. FontLab will analyse the Unicode information available in the font and will automatically detect which codepages this font can support.

To add a codepage to the list of supported codepages select a codepage in the left list and press the **Add** button.

To remove a codepage from the list of supported codepages select a codepage in the right list and press the **Del** button.

To reset the list of supported codepages, press the **Reset** button.

The Meaning of Supported Codepages

In Mac OS prior to 8.5 and Windows 3.1x this information is not used.

In Mac OS starting from 8.5, Windows 95 and Windows NT a font that has more than one standard (Mac OS Roman or 1252 Latin 1) codepage supported will appear as a font available for different scripts. So, if, for example, you set Latin 1 and Cyrillic codepages for a font with the name MyFont, in Windows 95 (and NT) it will appear as MyFont (Western) and MyFont (Cyrillic).

Type 1 Character Set

Type 1 fonts do not have such extensive support for multiple codepages. The character names they use to identify characters are mapped to codes through the encoding vector. There is one parameter that is used to tell Adobe Type Manager (used to support Type 1 fonts in Windows) how to interpret the encoding vector. This is the Microsoft Character set.

Some of the values for Microsoft Character set:

ANSI	The Font has all the characters necessary to represent the standard Windows Latin 1 character set. No reencoding is necessary.
Symbol	The font is symbolic with a custom encoding vector. It should appear as Symbol in Windows applications and the font's own encoding vector should be used to access characters.
ShiftJIS	This is a Japanese font that includes Kanji characters.
OEM	The font has MS DOS characters. This setting is rarely used in Type 1 fonts.
Bitstream	This is a normal text font, but it has its own encoding that should be used to access characters. This setting is highly recommended for all text fonts with a non-standard encoding vector.
Arabic	The font has Arabic encoding.

Other values cover more codepages that may be supported by the font. Choose the codepage that is the default for your font.

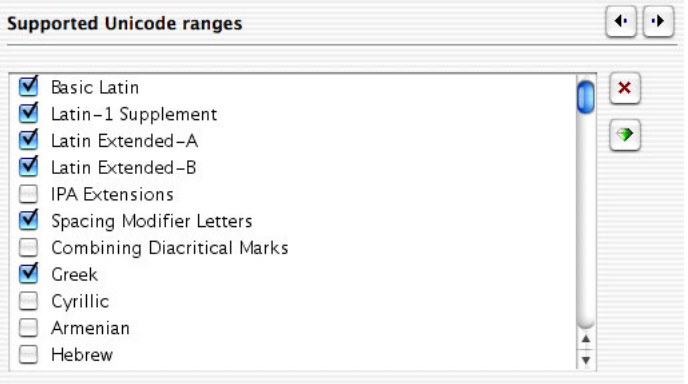
The Macintosh Script and FOND ID

A Macintosh script is used to calculate the correct value of the FOND ID parameter used in Macintosh font suitcases to differentiate fonts. The Script is necessary to properly identify a font in multiple-language non-Unicode programs.

You can use the list of all the possible Macintosh Scripts to automatically calculate the FOND ID in the correct range (FontLab will generate a random value) or you can manually enter the FOND ID and FontLab will automatically select the script related to that value.

You may change this field later when exporting your font in any of the Macintosh formats. And certainly this field is not needed when saving the font for Windows.

Supported Unicode Ranges



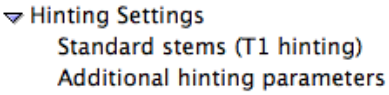
TrueType and OpenType fonts must declare Unicode ranges that the font can support so that the operating system can decide which characters to represent. In TrueType format this information is stored as the ulCodePageRange1 and ulUnicodeRange fields in the OS/2 table.

The Supported Unicode ranges dialog is relatively simple: you can see a list of all Unicode ranges with a check box to the left of each name. If the check box is checked it means that range is supported.

Buttons to the right of the list mean:

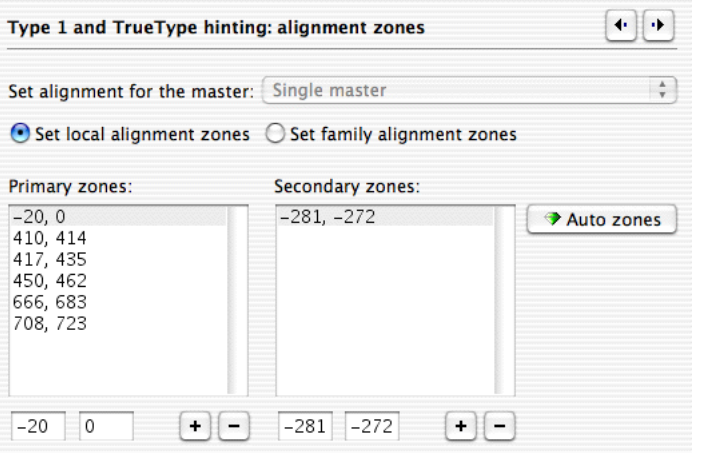
	Uncheck all ranges.
	Automatically check ranges using information about Unicode indexes assigned to fonts characters.

Hinting Settings



Information in this section is related to hinting, which is described in full detail in the Hinting chapter. The information below is copied there also, so if you're not interested in hinting right now, you can skip this section.

Alignment Zones



There are two list boxes where alignment zones can be set: the **Primary zones** list and the **Secondary zones** list.

In Type 1 terminology primary zones are called *BlueValues* and secondary zones *OtherBlues*.

BlueValues include one bottom alignment zone, the so-called *baseline zone,* and up to 6 top alignment zones. The baseline zone is used to control bottom overshoots that have to be aligned to the baseline.

OtherBlues includes up to 5 bottom alignment zones.

To add a new alignment zone, press the **Add** button below the list.

To edit the position of the zone, select the zone you want to edit in the list and edit it in the edit fields below the list.

To remove an alignment zone, select the zone you want to remove from the list and press the **Del** button below that list.

You can **see a preview of the zones** by switching on the Alignment Zones layer in the Glyph Window and pressing the **Apply** button in the FontInfo dialog box. You will see the alignment zones in the Glyph Window in light blue.

Press the **Auto zones** button to automatically calculate alignment zones in the **Primary zones** list box.

How FontLab Calculates Alignment Zones

To calculate alignment zones in the BlueValues list, FontLab finds characters with overshoots and characters that are flat in the position of the overshoot. Then it measures the top and bottom vertical positions of these characters and detects a zone. Examples of such characters are: 'o' and 'x', 'O' and 'H', 'p' and 'g', and so on. FontLab tries to find many different characters from different languages, so it is usually able to locate some examples.

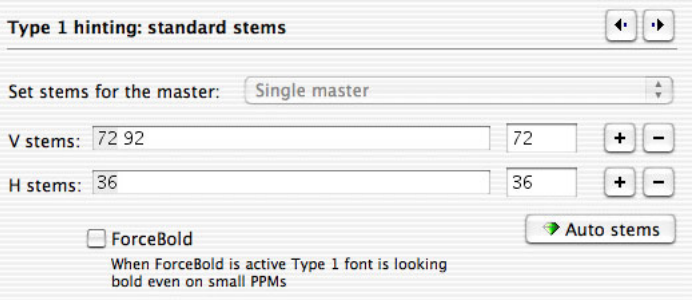
Alignment zones are also used in TrueType manual and automatic hinting.

Family Alignment Zones

To support the common appearance of fonts that belong to the same font family the Type 1 hinting system allows so-called FamilyBlues, alignment zones that are used in the whole font family.

To set family alignment zones switch on the **Set family alignment zones** check box. Then edit the alignment zones as usual. To return to editing "local" alignment zones switch on the **Set local alignment zones** check box.

Type 1 Standard Stems



Standard stem widths are also controlled through the **Alignment Zones** page of the **Font Info** dialog box. To edit standard stems open the Font Info dialog box and select the **Alignment Zones** page.

There are controls for both vertical and horizontal standard stem widths. All available horizontal stems appear in the horizontal lists. You can select any stem just as you would in normal, vertical list controls.

To add a stem to the list of the standard stems press + at the right side of the stems' list.

To edit a standard stem width, select it using the mouse button and edit its value in the edit field to the right of the list.

To remove a stem from the list, select it and press the – button.

Note that FontLab will sort stem widths starting from the second in ascending order when you close the **Font Info** dialog box.

ForceBold – when this option is switched on, Type 1 rendering algorithm makes the font looking "bold":

The quick brown fox ForceBold is OFF
The quick brown fox ForceBold is ON

StdHW, StdVW, StemSnapH and StemSnapV Parameters

From the Type 1 font specification you may know that in Type 1 fonts two types of standard stem widths are used: Standard Width and Stem Snap Width. There is one standard width for each direction and up to 10 stem snap values. In FontLab these values are united in the stem list. StdHW and StdVW are taken from the first records in the stem lists. StemSnapH and StemSnapV records are the remaining records in the stems' list.

FontLab also has a faster way to append stems to the list of standard stem widths. Any vertical or horizontal hint may be used as a source of stem width information. Just point the Edit tool at the hint, **Ctrl**-click mouse button and select the **Define a Stem** command in the popup menu. If this command is not accessible, it means that this stem is already in the list.

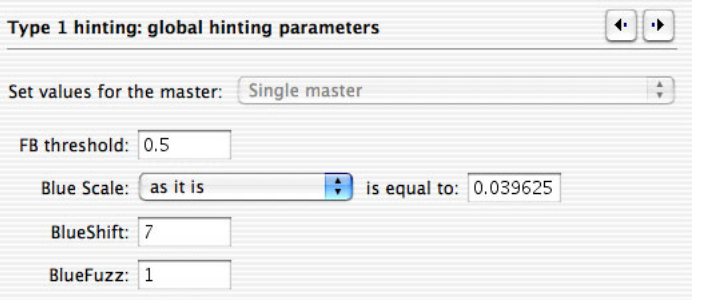
To automatically calculate standard stem widths press the **Auto stems** button.

How FontLab Calculates Standard Stems

FontLab can calculate standard stem widths only if some characters in the font have Type 1 hints.

1. It builds a table of all hints that are used in the font, sorts this table by frequency of usage and selects the most frequently used hints.
2. All selected hints are then compared with these most frequently occurring stem widths and hints with widths that are close together are combined into a single record.
3. The list is then sorted again.
4. The most frequently used elements are then selected and used as standard stems.

Global Hinting Parameters



Some additional data may be set to control the hinting process:

FB threshold	ForceBold threshold – this parameter is used only in Multiple Master font and controls when the Force Bold parameter is ON in an intermediate design
BlueScale	Controls PPM when overshoot depression is switched off
BlueShift	Gives more precise control over overshoot description (see below)
BlueFuzz	Expands alignment zones in both directions.

You can set all these values in the right part of the **Alignment Zones** page of the **Font Info** dialog box.

BlueScale is the PPM size at which overshoot suppression is switched off. If PPM is less than BlueScale, then overshoot suppression is applied. If it is equal to or exceeds BlueScale, overshoot suppression works only if the distance from the aligned point to the base line of the alignment zone is less than the **BlueShift** value and the scaled distance is less then half of a pixel.

The BlueScale value is stored in Type 1 fonts in a very strange format, but in the **Alignment Zones** page you can set it using one of three different ways: directly, in the form that the Type 1 specification describes (i.e. it looks like a floating point number), as a PPM size, or as a point size on a device with 300 DPI resolution. Use the **BlueScale:** popup to select the BlueScale editing method and edit it in the **is equal to:** edit field.

BlueScale Formulas

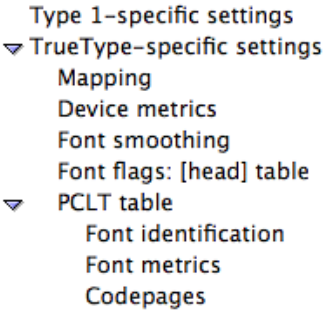
The "actual" value of the BlueScale value is calculated as:

$$BlueScale = \frac{\left(PPM - 1.63333\right)}{800.0}$$

The **BlueFuzz** value allows you to expand the action range of the alignment zones in both directions. Thus if you have defined a zone like (700-715), and BlueFuzz is equal to 2, then the actual zone used will be (698-717). This is usually used when you are not sure that you correctly set all the alignment zones or when the characters are not all precisely aligned. The normal value of this parameter is 0 but by default it is set to 1.

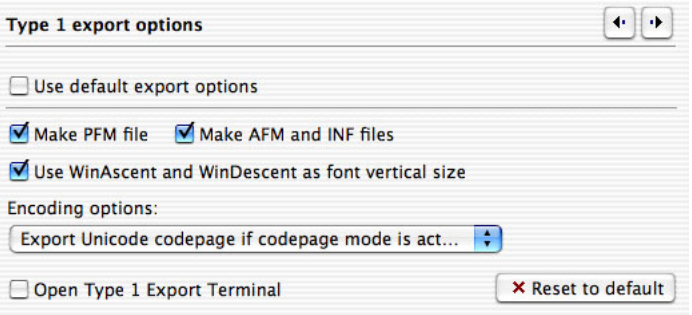
TrueType hinting algorithms do not use BlueScale, BlueFuzz and BlueShift values.

Format-Specific Options



This section covers options that are specific to Type 1 and TrueType/OpenType fonts. For both destination formats you may choose font-specific export options that will be used instead of globally set FontLab options. In addition, for TrueType fonts you can customize some settings that are necessary for high-quality rendering.

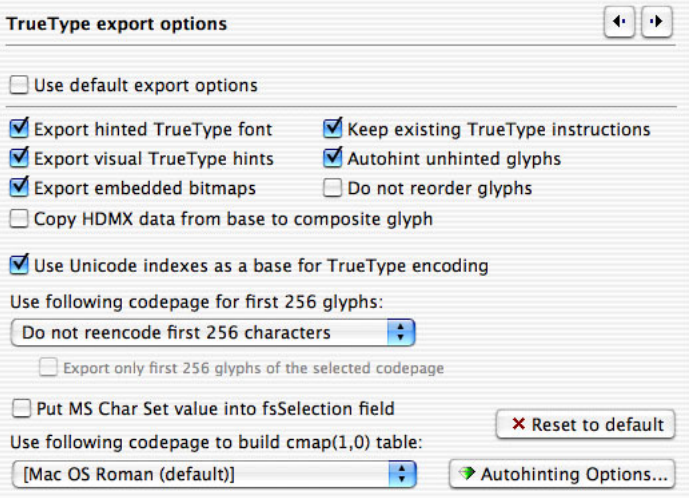
Type 1 Export Options


Type 1 export options
Use default export options
Make PFM file
Make AFM and INF files
Use WinAscent and WinDescent as font vertical size
Encoding options:
Export Unicode codepage if codepage mode is act...
Open Type 1 Export Terminal
Reset to default

Use this page to customize font export options for the current font. This page copies Preferences/Type 1 export settings, so you may refer there for the detailed description.

Switch on the **Use default export options** parameter to use default export options that are not font-specific.

TrueType Export Options


TrueType export options
Use default export options
Export hinted TrueType font
Keep existing TrueType instructions
Export visual TrueType hints
Autohint unhinted glyphs
Export embedded bitmaps
Do not reorder glyphs
Copy HDMX data from base to composite glyph
Use Unicode indexes as a base for TrueType encoding
Use following codepage for first 256 glyphs:
Do not reencode first 256 characters
Export only first 256 glyphs of the selected codepage
Put MS Char Set value into fsSelection field
Reset to default
Use following codepage to build cmap(1,0) table:
[Mac OS Roman (default)]
Autohinting Options...

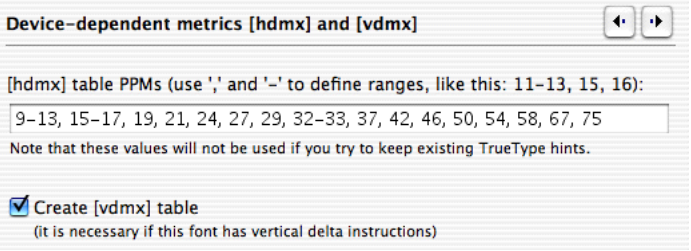
Use this page to customize font export options for the current font. This page copies Preferences/TrueType export settings, so you may refer there for the detailed description.

Switch on the **Use default export options** parameter to use default export options that are not font-specific.

TrueType Mapping Settings

If the **Automatically Add...** option is on, then FontLab will analyse the font and add characters that are necessary for complete font compatibility. In Windows only two characters are really necessary: the ".notdef" and "space" characters. On the Macintosh a couple additional characters are required: "CR" and "NULL". Note that FontLab will generate these characters only if they do not exist in the font, so you can create them manually and control their appearance. We recommend leaving this option on, especially if you are developing Macintosh fonts.

Device-Dependent Metrics


Device-dependent metrics [hdmx] and [vdmx]
[hdmx] table PPMs (use ',' and '–' to define ranges, like this: 11–13, 15, 16):
9–13, 15–17, 19, 21, 24, 27, 29, 32–33, 37, 42, 46, 50, 54, 58, 67, 75
Note that these values will not be used if you try to keep existing TrueType hints.
Create [vdmx] table
(it is necessary if this font has vertical delta instructions)

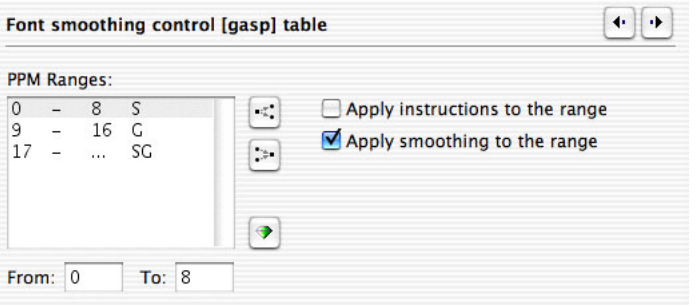
The **[hdmx]** control lets you customize the sizes for which FontLab will generate records in the hdmx TrueType table. This table is used to pre-calculate pixel metrics of font glyphs so it will not be necessary to run a hinting program to get the correct width.

If you don't want FontLab to generate an hdmx table for your font, empty this editing field.

Use a colon to separate entries and '-' to define ranges of PPM.

Use the **Create VDMX Table** control to ask FontLab to automatically calculate a VDMX (Vertical Device Metrics) table. This is necessary if some characters in the font are hinted in the vertical direction, so that at some resolutions they can extend above or below the scaled Win/Mac Ascender or Win/Mac Descender values and unwanted dropouts do not appear. We strongly recommend you always have this option switched on.

Font Smoothing Control



To improve the appearance of TrueType fonts on the screen the latest versions of the Mac OS and Windows operating systems use a special technique called font smoothing. With this technique edges of the characters are rendered using shades of gray:

Visually this decreases the dither of the characters' edges so that text is easier to read. This technique may be combined with gridfitting methods that optimize the character's appearance by adjusting its outline.

The font smoothing options let you control when to use one or both of these techniques.

In the list box you see PPM (Pixels Per eM — font size measured in screen pixels) ranges with one or two letters describing the applied technique.

S means that smoothing will be applied; **G** means that gridfitting will be applied; **SG** means that both techniques will be combined to achieve the best results.

To set options for a range, select the range and change its settings using the controls in the dialog box. Use the check boxes to select the rendering technique and the edit controls below the list to change the PPM range.

To define a new range, select one of the existing ranges and press the **Split** button. The range will be split into two ranges and you will be able to set different options for them.

To merge two ranges select a range and press the **Merge** button. The selected range will be merged with the range that is above it.

To reset the font smoothing settings press the **Auto** button and they will be calculated as advised by the TrueType specification.

Font smoothing on Macintosh

The [gasp] table is not used when fonts are rendered on Macintosh . So you will not see the results of these settings in the TrueType hinting Preview window when the TrueType hinting tool is active. Refer to the "Hinting" chapter for more information.

[head] Table Settings

These entries are specific for TrueType rendering. We do not recommend you modify them if you are not sure of what you are doing.

LowestRecPPem	This parameter controls the lowest PPM at which the TrueType rasterizer is allowed to apply instructions
FontDirectionHint	This [head] table parameter defines font direction, which may have the following values: 0: Fully mixed directional glyphs 1: Only strongly left to right 2: Like 1 but also contains neutrals -1: Only strongly right to left -2: Like -1 but also contains neutrals. A neutral character has no inherent directionality. Spaces and punctuation are examples of neutral characters.

Basic PCLT options

The PCLT table is used when a TrueType font is printing on a PCL printer. PCL (Printer Control Language) is a page-definition and printer-control language developed by Hewlett Packard and is used in their printers.

This table is recommended for TrueType fonts, but only if you understand all the options and parameters.

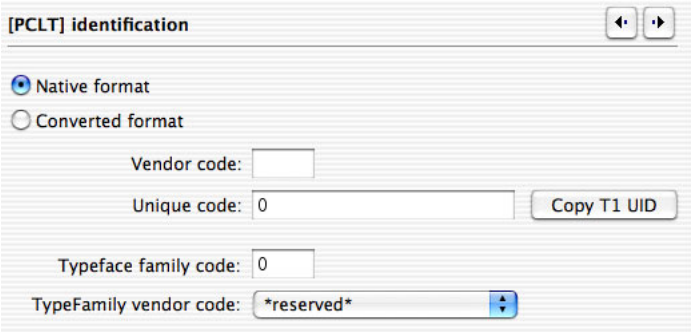
The first page includes only one option: **Export the PCLT table,** which controls table export. Note that you can define table parameters but not export the table – all values will be saved to your FontLab font file (VFB).

Extra information on many of PCLT fields can be found in the *HP PCL 5 Printer Language Technical Reference Manual* available from Hewlett-Packard Boise Printer Division.

Our description of various PCLT fields is partially copied from the PCLT table specification available online at:

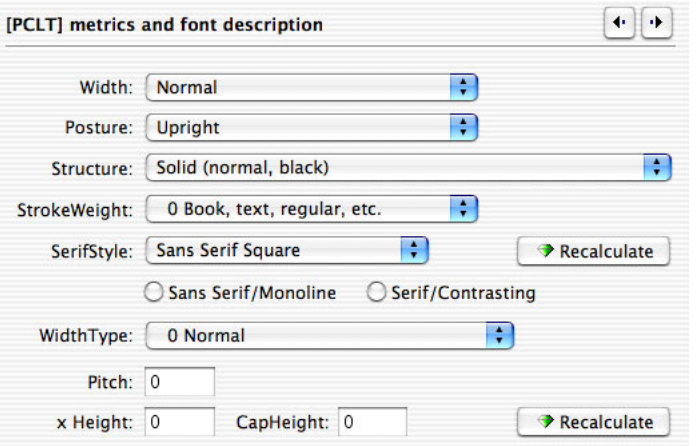
http://www.microsoft.com/typography/otspec/pclt.htm

PCLT Identification



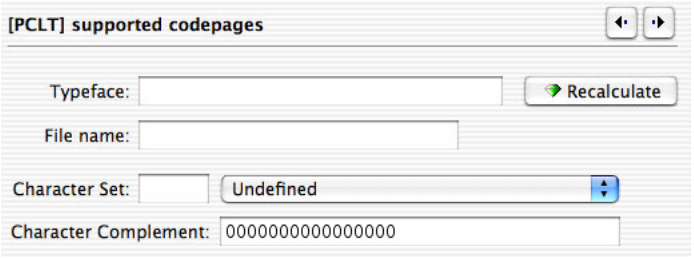
Native/Converted format	Only original font vendors can select Native format. If you are modifying an imported font, select Converted format.
Vendor code	Single-character code of the font vendor assigned by Hewlett-Packard Boise Printer Division to major font vendors. Below is a list of registered font vendors: A Adobe Systems B Bitstream Inc. C Agfa Corporation H Bigelow & Holmes L Linotype Company M Monotype Typography Ltd.
Unique code	Font's unique code – 24-bit integer. Using the Copy T1 UID button you can copy the Type 1 Unique ID value into this field.
Typeface family code	This is a family code assigned by HP Boise Division. Refer to HP manuals for more information.
TypeFamily vendor code	This is another vendor code assigned by the HP Boise Division. Choose one of values in the list.

PCLT Metrics and Font Description



Width	The appearance width. Select one of the options in the list.
Posture	The slant of the glyphs. Could be upright, italic/oblique or reversed italic.
Structure	The structure of the glyph's interior. Select one of the options in the list.
StrokeWeight	Weight of the font. This value is related to the **Weight** parameter on the **basic names** page.
SerifStyle	Style of the glyph's serifs. This is related to the PANOSE settings. Use the **Recalculate** button to calculate values automatically using information from other pages.
WidthType	PCL appearance width value. Select one of the options in the list.
Pitch	The width of the space in font units.
xHeight	The height of the optical line describing the height of the lowercase x in font units. This might not be the same as the measured height of the lowercase x.
CapHeight	The height of the optical line describing the top of the uppercase H in font units. This might not be the same as the measured height of the uppercase H.

PCLT Codepages



Typeface

This 16-byte ASCII string appears in the "font print" of PCL printers. Care should be taken to insure that the base string for all typefaces of a family are consistent, and that the designators for bold, italic, etc. are standardized.

Times New
Times New Bd
Times New It
Times New BdIt

File name

This 6-byte field is composed of 3 parts. The first 3 bytes are an industry standard typeface family string. The fourth byte is a treatment character, such as R, B, I. The last two characters are either zeroes for an unbound font or a two-character mnemonic for a symbol set if a symbol set is found.

Examples:

TNRR00 Times New (text weight, upright)
TNRI00 Times New Italic

Character Set

HP Boise Division assigns symbol set values. Unbound fonts, or "typefaces" should have a symbol set value of 0. See the *PCL 5 Printer Language Technical Reference Manual* or the *PCL 5 Comparison Guide* for the most recent published list of codes.

In most cases all you need to do is to select one of pre-defined values in the popup menu to the right of the manual field. See below for more information.

Character Complement

This 8-byte field identifies the symbol collections provided by the font, each bit identifies a symbol collection and is independently interpreted. Symbol set bound fonts should have this field set to all F's (except bit 0).

File Name Treatment Flags:

R	Text, normal, book, etc.
I	Italic, oblique, slanted, etc.
B	Bold
J	Bold Italic, Bold Oblique
D	Demibold
E	Demibold Italic, Demibold Oblique
K	Black
G	Black Italic, Black Oblique
L	Light
P	Light Italic, Light Oblique
C	Condensed
A	Condensed Italic, Condensed Oblique
F	Bold Condensed
H	Bold Condensed Italic, Bold Condensed Oblique
S	Semibold (lighter than demibold)
T	Semibold Italic, Semibold Oblique

Character Set Table

FontLab stores the list of pre-defined character set codes with descriptions in a special text file named pclset.dat located in the FontLab/Data folder.

This file has a simple structure:

-- Undefined
19M Adobe Symbol
0V Arabic (McKay's)

Every line contains a Character set code followed, after a single space, by a description of the character set. You can open this file in any text editor, like SimpleText, and make any changes you want.

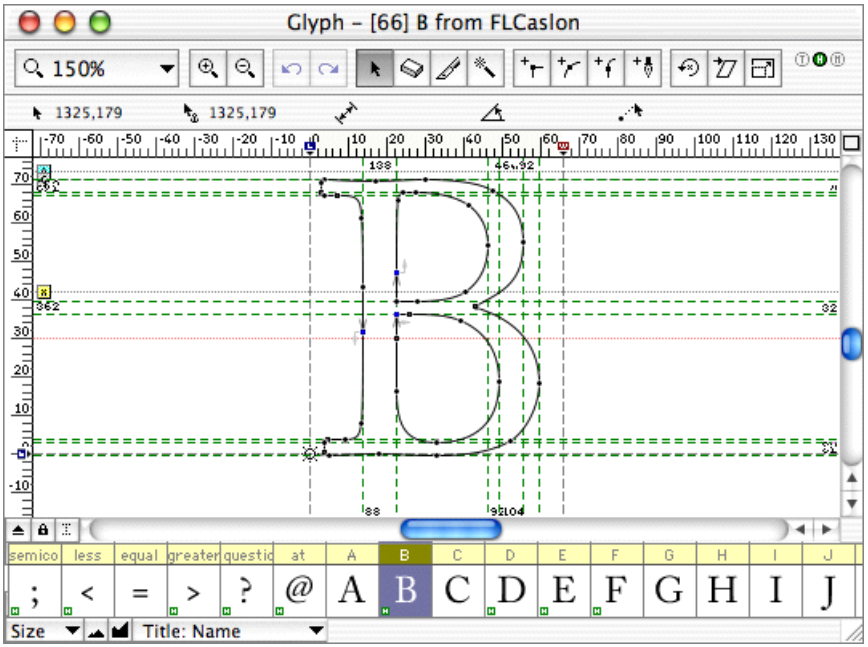
The Glyph Window

The Glyph Window is a standard tool in all FontLab based applications. It is a universal and very powerful contour-editing module that also allows you to perform many font-specific operations.

4

Glyph Window Contents

Open the Glyph Window by double-clicking any character sample in the Font Window or Metrics Window.



The Glyph Window

The Glyph Window has the following parts:

- Toolbar area
- Toolbar area expand/collapse button
- Control marks area
- Editing Field
- Top and left rulers
- Left-Top box
- Scroll Bars
- Glyphs Bar
- Glyphs Bar expand/collapse button
- Lock button
- Meter panel button
- Note icon area
- Glyph mark area

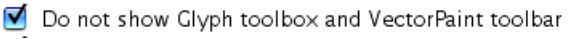
The *Local Toolbar* is the command center of the Glyph Window. You can switch it on and off by pressing the small button at the top of the vertical scroll bar: . There is a popup menu with zoom selection and two pairs of buttons:

The first pair of buttons is used to select the zoom mode.

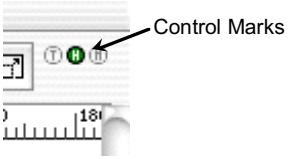
The second pair of buttons is used to apply **Undo** and **Redo** operations.

Some editing tools (which we discuss later) may have their own toolbars. The collection of common tools that are available in the Edit and Sketch modes and reside in the local toolbar is called the *Glyph Toolbox*:

The glyph toolbox is invisible by default. To make it visible uncheck the option in the Glyph Window page of the Preferences dialog box:



There is a control marks panel in the right-top area of the local toolbar.



This panel duplicates some of the information that appears in the character cell of the Font Window so that you can instantly get important information about the current glyph.

Three icons may appear in this area:

	The glyph has either imported or visual TrueType hints
	The glyph has a Type 1 hint replacement program
	The glyph has a Mask layer that is “compatible” with the outline.

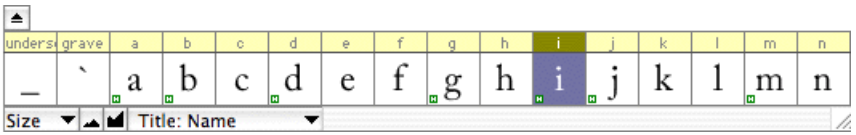
Below the Local toolbar and at the left of the window you may see rulers that are used to preview positions of various structures in the glyph space. You may switch the rulers on and off with the **View/Rulers** command or using the context menu that appears if you **Ctrl**-click one of the rulers.

In the bottom-left corner of the Glyph Window you will find a little button . Click it and you will see the Glyphs Bar – a small subset of the Font Window that lets you quickly browse the font or select characters for editing.

Let’s stop for a moment and describe the Glyphs Bar.

The Glyphs Bar

To open the Glyphs Bar you click on the Glyphs Bar expand button or press the **B** key when the Glyph Window is active.



If the Glyphs Bar is open, you use the **Tab** key to switch between the editing area and the Glyphs Bar. When the Glyphs Bar has focus, it is slightly highlighted.

To customize the Glyphs Bar's appearance, use the **Size** popup menu, **Zoom In** and **Zoom Out** buttons in the footer. Use the **Title** popup menu to control the cells' caption:

None	To hide the cell's caption
Name, Unicode, Index, Width, Decimal, Hex, Octal, ANSI	Choice of information to show in the caption. Options are the same as the choices in the Font Window.

To select a glyph for editing double click the glyph cell. Or, if the Glyphs Bar is active, choose the cell with the left and right arrow keys and click the **Enter** key.

If you hold down the **Shift** key while using the left and right arrow keys glyphs are opened instantly.

The fastest way to open a glyph is to use the keyboard:

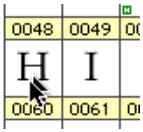
1. Set the focus to the Glyphs Bar with the **Tab** key.
2. Press the key of the character that you want to open on the keyboard.
3. If the glyph is not directly accessible from the keyboard, enter its name. You have one second to enter each character of the name.
4. Click the **Tab** key again to return focus to the editing area so you can edit the outline.

You may close the Glyphs Bar at any time by clicking the Glyphs Bar collapse button . When the editing area is active, you can just press the **B** key.

Selecting a Glyph for Editing

In addition to using the Glyphs Bar you can open a glyph in the Glyph Window using any of the following methods:

- Double click the glyph's cell in the Font Window to open it:



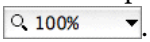
  If you already have an open Glyph Window with a glyph from the same font, the new glyph will be opened in the same Glyph Window (replacing the previous glyph). Hold the **Cmd** key down when you double-click on the glyph cell to open it in a separate Glyph Window. Note that if this method doesn't work it usually means that it is switched off in the Font Window page in the Preferences dialog box.

  You can force FontLab to **always open a glyph for editing in a new Glyph Window**. Use the **Create new window by double-click** option in the Font Window page to activate this feature.

- **Ctrl**-click in the Font Window and select the **Open in New Glyph Window** command to open the glyph in a separate Glyph Window.
- Click on this button in the bottom left area of the Glyph window to unlock the keyboard and click on the keyboard to open the character for editing. You can also type the glyph name if you know it. Use the **Cmd-L** shortcut to quickly lock and unlock the keyboard.
- Click the mouse button on the glyph selected in the Font Window and drag it into any Glyph Window.
- Press the or buttons in the Tools or Edit toolbar to move to the previous or next glyph. The Keys ',' and '.' are equivalents for these buttons.
- Select the **Choose Glyph** command in the **Glyph** menu and find the glyph that you want to open.

Changing the View in the Glyph Window

Use the zoom mode and scroll bars to change the view in the editing field of the Glyph Window. By using the scroll bars you can scroll the viewing field of a symbol. With the zoom mode you can define how the symbol coordinates are converted to screen coordinates and vice versa. If you choose a bigger zoom mode you will see a more detailed symbol and you can do the editing operations more precisely. However, in the larger zoom modes not all of the symbol will lie inside the editing field so you will have to use the scroll bars to see the different parts of the symbol.

There are fixed zoom modes and custom zoom modes. You can select one of the fixed zoom modes in the **Zoom** popup menu located in the upper part of the Glyph Window: 

. When you choose a fixed zoom mode FontLab will return to this glyph mode on every **Zoom Out** command.

To magnify part of the glyph, select the **Zoom** tool (button on the toolbar or the + key on the keyboard) and declare a custom zoom mode. This mode is temporary and you can always return to the previously selected fixed zoom mode by clicking on the button (or by clicking on the - key).

Alternative keyboard shortcuts are:

Command+Space, then click	Zoom in
Command+Option+Space, then click	Zoom out
Command+1,2,3,4,5,6	Set fixed zoom mode from 6.25% to 200%
Space and drag	Scroll (hand cursor appears)

After you select the zoom tool , move the mouse pointer to one of the corners of the rectangular area that you want to zoom on and press the mouse button . Then, holding the mouse button down, define the zoom-in area by dragging the cursor to form a rectangle. Release the button and the new zoom mode will be selected.

Quick Zoom Selection

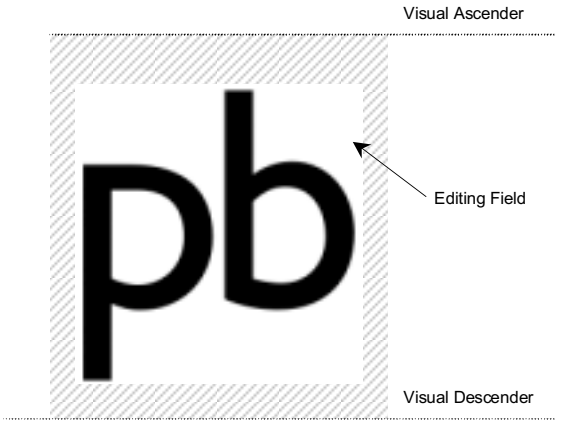
You can quickly change the zoom mode of the Glyph Window by selecting the **Zoom In** or **Zoom Out** command from the **View** menu. Alternatively you can click the **Z** key for zooming in or the **X** key for zooming out.

This command increases or decreases the zoom mode by a factor of two. If the mouse cursor is in the editing area of the Glyph Window the new zoom mode will be centered around the cursor position.

These keys are active even when you drag something with one of the editing tools.

Vertical Alignment Options

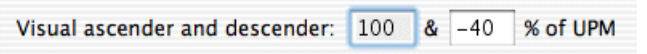
When you select 100% as the zoom value, FontLab needs to choose a scaling factor to fit the font unit space in the Glyph Window. Two vertical levels in the font space define this scaling: Visual Ascender and Visual Descender:



When you select 100% zoom, it means that Visual Ascender is fitted to the top of the editing field and Visual Descender to the bottom.

The same values are used to build the icons that you see in the Font Window or panels like Collection/Classes.

To set Visual metrics, use the **Glyph Window** page of the **Preferences dialog box** (**Application/Preferences** menu):



Values are measured in percentage of font UPM, so –20% is –200 if UPM is 1000 and –410 if UPM is 2048.

Tools and Operations

FontLab's Glyph Window may work in several modes. The four most important modes are:

	Edit mode	The main mode used to move everything in the glyph, from guidelines to nodes to glyph margins
	Sketch mode	Used to draw new outlines
	VectorPaint mode	A set of tools used to create new glyphs or modify existing glyphs
	Meter mode	Used to measure contours.

These modes are represented by the first four buttons in the Tools toolbar (which can be switched on and off in the **View/Toolbars** menu).

Other modes include TrueType and Type 1 hinting and eight additional operations:

	Transform Selection	Scales, rotates or skews the selected portion of the outline
	Interpolate	Modifies the outline by moving a few "base" points
	Background	Sets the size and position of the bitmap background layer
	Envelope	Modifies the outline as if it was stretched on rubber
	Curve	Approximates a segment of the outline with a curve
	Set	Lets you set precise positions of outline points
	Reverse contours	Reverses the contour's direction
	Set startpoints	Changes the starting points of contours and rearranges contours
	Type 1 hinting	Type 1 hinting
	TrueType hinting	Manual TrueType hinting

These buttons are available in the Tools toolbar and separately in the Operations toolbar (which can be switched on and off in the **View/Toolbars** menu).

When the Glyph Window is switched to one of these modes, you may see additional toolbars, panels and dialogs. They are shown and hidden automatically, depending on the mode and most of them hide when the Glyph Window is deactivated.

Use the buttons on the Tools toolbar or keyboard shortcuts **to switch modes**. The four most important modes are:

OPTION+1	Edit mode
OPTION+2	Sketch mode
OPTION+3	VectorPaint mode
OPTION+4	Meter mode

Edit Mode

The **Edit** mode is the most important mode in FontLab. In this mode you can modify the contents of all the editing layers.

All operations performed with the edit tool can be undone with the **Undo** command of the **Edit** menu, or just by clicking the **Undo** button on the toolbar at the top of the Glyph Window. You can undo up to 200 operations. All undone operations can be redone with the **Redo** command of the **Edit** menu or with the **Redo** button on the toolbar.

In Edit mode you can use eleven different Edit Tools. You can easily choose one of the tools using the Edit toolbar or the Glyph toolbox:

Alternatively you may use the keys from 1 to 8 to quickly select edit tools:

	1 Edit	Main tool, used to drag objects on the editing layers and perform other operations. In the following chapters we will assume that this tool is active in the Edit mode.
	2 Eraser	This tool is used to quickly remove unnecessary nodes.
	3 Knife	Tool to insert nodes and break outlines.
	4 Magic Wand	Tool to quickly select contours (just click anywhere near the contour and it is selected).
	5, Add Corner, **6, Add Curve,** **7 Add Tangent**	Tools to create new contours or insert nodes.
	8 Bezier Drawing	Tool to draw the contour with the Bezier curves.
	Rotate, **Slant,** **Scale**	Tools to quickly transform outlines.

The Glyph toolbox is available if you enter the Edit mode. If you don't want it to appear, use this option on the Glyph Window page of the Preferences dialog box:

When this option is active, the Glyph toolbox will be hidden, so you will have to use keyboard shortcuts to access the different tools of the Edit mode. You can also use the comfortable Edit toolbar available in the **View/Toolbars** menu.

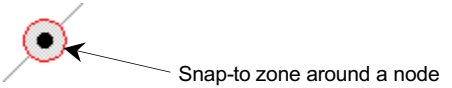
Temporary Activating the Edit Tool

You can temporary activate the Edit tool from any other tool. Just click on the **Cmd** key on the keyboard. Second click on the **Cmd** key will return the tool you were using before.

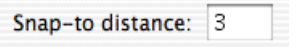
Snap-to Distance

In the following sections we will discuss how to use the Edit tools to modify the outline and other editing layers. All other tools will be explicitly named.

When you need to select a node or any other object on any of the layers, you need to click it with the mouse. You don't need to click the object precisely, but you must be within a certain distance, which is called the "snap-to distance".



Snap-to is used when you select an object for which the feature is allowed. By default the snap-to distance is set to 3 screen pixels, but you can change it on the **Glyph Window** page of the Preferences dialog box:



Editing Layers

In FontLab every glyph contains several editing layers. Some of them are used when the font is exported, others are FontLab-only and are used to help you work with the glyph. Below is the list of all the layers that you can see in the Glyph Window. Later we will describe them in full detail.

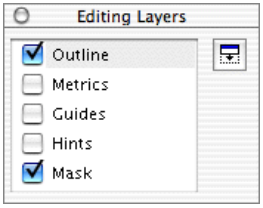
	Outline	Main layer containing the glyph's outline. There is no button to switch the outline layer off.
	Grid	Regular grid which helps to align the outline
	Guidelines	Horizontal, vertical and/or diagonal guidelines
	Hints	Type 1 hints – pairs of two vertical or horizontal lines set at a fixed distance
	Mask	Outline template
	Background	Bitmap background
	Alignment zones	Special zones that define the overshoot positions in the font
	Glyph metrics	Glyph metrics – left and right sidebearings and a baseline
	Vertical metrics	Font's vertical metrics, such as ascender, descender or cap height
	Template	Global (font-wide) outline template
	Anchors and carets	Anchors are special named marks and carets that define stop positions in ligatures

You can control the layers' appearance and features with the **View** menu:

Show layers	Lists all layers (and a few other options) and lets you switch them on or off. You cannot switch off (hide) the outline layer.
Lock layers	Lists the layers that you can lock to protect from accidental modification.
Snap to layers	Controls which layers have the "snap-to" feature activated.

Editing Layers Panel

An alternative way to control the editing layers features is the special Editing Layers panel which you can put on the screen with the **Window/Editing Layers panel** command (or with this button on the Panels toolbar):



Please note that you must open the Glyph Window to make it possible to use the Editing Layers panel.

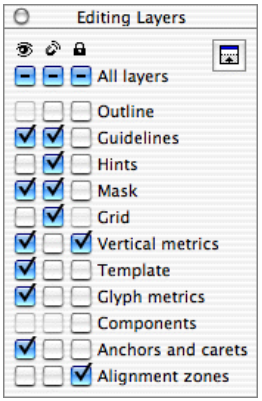
To activate the layer, just select it in the list. When you do it, this layer becomes unlocked and all other layers are locked. If activated layer was hidden, it will appear until it is deactivated.

Use check boxes to the left of the layer names to control layer appearance.

For example, **to edit the glyph metrics**, just select the “Metrics” in the list. Glyph and font metrics layers will appear in the glyph window and will be available to edit while all other layers (outline, hints, guides and mask) will be locked.

To edit the Mask layer, just select “Mask” in the list. The outline layer will be shown as a mask (and will be locked), hints will disappear and the mask layer will be set to front and open for editing.

The Editing Layers panel may be opened in the expanded mode. Use this button to switch the panel between simple and expanded mode:



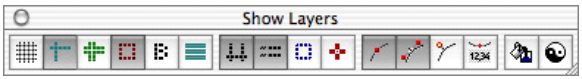
The left column of checkboxes controls the layers' appearance in the Glyph Window.

The middle column controls the "snap to" feature and is equivalent to the **View/Snap to layers** menu.

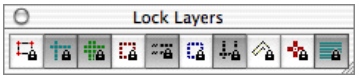
The right column lets you lock layers and is equivalent to the **View/Lock layers** menu.

With the panel you can easily control all the layers at once using the check boxes in the top row:

If you frequently need to switch some layers on and off you can use the **View/Toolbars** menu to open the **Show Layers** toolbar:



The same operation is possible with the **Lock layers** toolbar:



In the following sections we will describe all the editing layers and their modification and control.

Outline Layer

The Outline layer is the most important of all the layers. It stores information about the glyph shape while all the other FontLab editing layers and most of the tools are designed to help you create good outlines. Before we turn to the outline editing tools let's talk about outline structure.

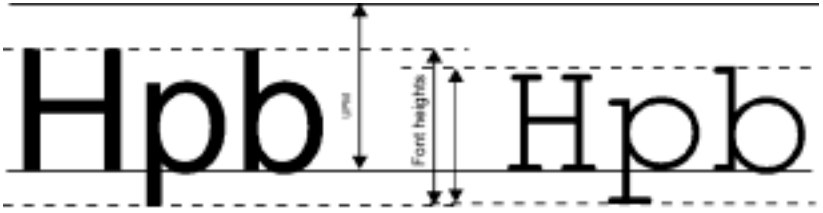
Units of Measurement

The coordinates of any object in the font are presented in a standard measurement system. One unit of this system is called a font unit. The scale of font units used in a particular font is the *Units Per eM (UPM)* font parameter. The program that scales the font knows the UPM value of the font and can use it to properly scale it. To get a text string of the same visible size a font that has a larger UPM value must be scaled with a smaller scale factor.

Usually Type 1 fonts have a UPM value of 1000 font units. Therefore, to get a text string with a height of 100 pixels (assuming that we use a raster output device) we would scale this font with a scale factor of 10%.

TrueType fonts usually have a UPM of 2048 units. So to get a text string of 100 pixels we would scale this font with a scale factor of 4.9%. Essentially the UPM is nothing more than a base value used for proper font scaling and as a base value for the font unit.

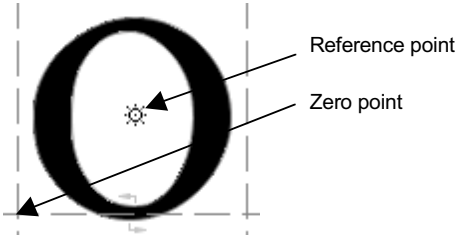
A more “graphical” font parameter is the *font height*. The font height (measured in font units) is the measurement of the font that is used to align strings in text. It is important not to confuse UPM and font height. UPM is just a scaling base, and, for example, all Type 1 fonts have the same UPM of 1000. Font height depends on the font design and may be different:



The font height can be defined as the distance from the bottom of a letter that is partially located below the baseline, like the ‘p’ character, and the topmost point of an uppercase character, like ‘H’, or a tall lowercase character, like ‘b’. Sometimes a font contains special glyphs that can be taller than ‘b’, like an integral sign, but usually these glyphs are not counted when font height is measured.

Reference Points

By default all coordinates are measured relative to the *zero point* of the glyph. This is located at intersection of the baseline and the left sidebearing line:



As an alternative, distances may be measured relative to the *reference point*, which may be positioned by the Edit tool to any point in the glyph space. Often a reference point is very useful when you are working on a symmetrical shape.

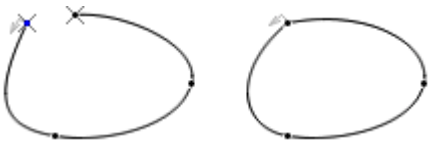
By default the reference point is located on the position of the zero point.

Contours

The most important and most complex information in a font is the glyph's shape. All glyphs are defined as a series of contours. All contours consist of a series of graphical primitives: straight lines and curves. Nodes – points that know their type – define all graphical primitives.

Open and Closed Contours

Contours may be open or closed:

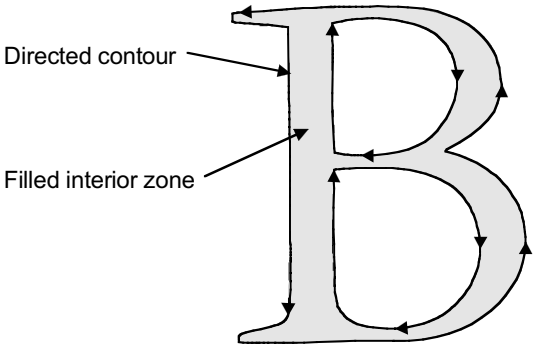


All known font formats require contours to be closed, but during outline editing it may be useful to have some contours in an open form and later connect them to each other to build final closed contours.

In FontLab it is very easy to open closed contours or to close open contours. It is also possible to customize the appearance of the open contours: they may be automatically filled (they are automatically closed by an invisible straight line that connects their starting and ending points) or not and their starting and finishing point may be optionally highlighted.

Filled and Unfilled Contours

Contours can be of two types: black or white. They can also be of two directions: clockwise or counterclockwise. The basic rule that applies to Type 1 fonts is simple: clockwise-directed contours are white and counterclockwise contours are black. A simpler form of the rule, known as the rule of the *left hand*, is: if you face along the direction of a contour, black (fill) will be on your left side.



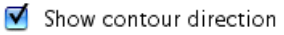
In the TrueType specification the opposite is the case, so a contour is filled on the right hand side. However, not all TrueType rasterizers require glyphs to follow this rule, so it is recommended, but not necessarily required, that you reverse contour directions when you are converting Type 1 fonts to TrueType.

Startpoint and Closepath

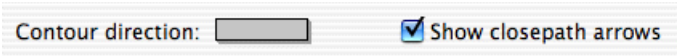
All contours have a *startpoint*. The startpoint is the first node of the contour. The last node of the closed contour is automatically connected to the startpoint with a straight line, which is called *closepath*. The color of the start point in the Glyph Window is **blue**.

Optionally, a contour-direction mark may appear on a startpoint:

You can switch this direction mark off with the option in the **Application/Preferences/Glyph Window**:



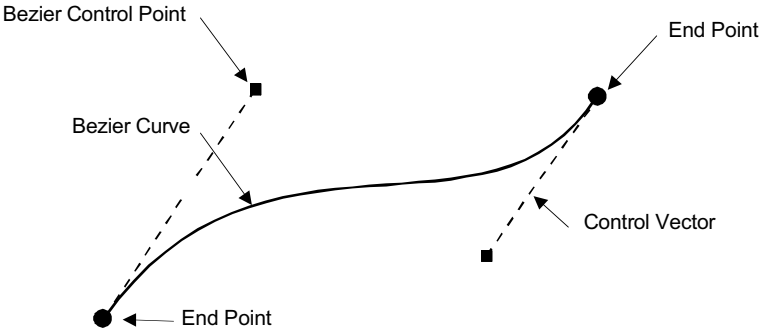
You can customize the appearance of the closepath with the Colors dialog box accessible with the **Colors...** button on the same page:



Contour direction lets you specify the color of the contour direction mark (so you can make it almost invisible, but still know the direction of the contours) and **Show closepath arrows** controls rendering of the closepath: if this option is active, closepath appears as an arrow:

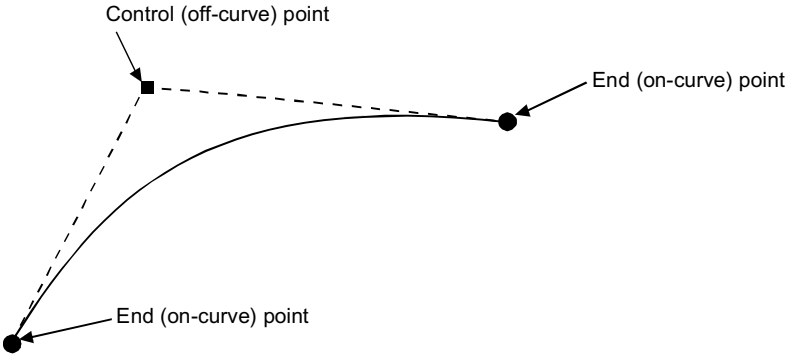
Curves and Lines

Graphical primitives are of three types: straight lines, Type 1 curves or TrueType curves. Straight lines (we'll call them *vectors*) are just straight lines that connect two sequential nodes. Type 1 curves are *Bezier curves* (3rd order, cubic b-splines). To modify the form of the curves two additional sub-nodes are used:

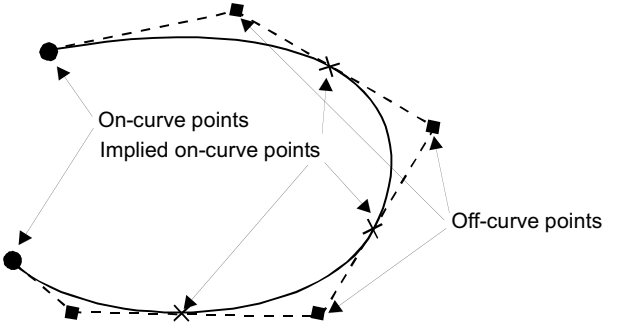


These *sub-nodes* are called *control points* and the vectors that connect the control points with the curve's ends are called *control vectors*. In the Glyph Window vectors (straight lines) end in **square** (in black-white mode) or **red** (in color mode) dots and curves end with **round** or **green** dots.

TrueType curves are 2nd-order curves (quadratic *b-splines*) that have one control point, called the "*off-curve*" point:



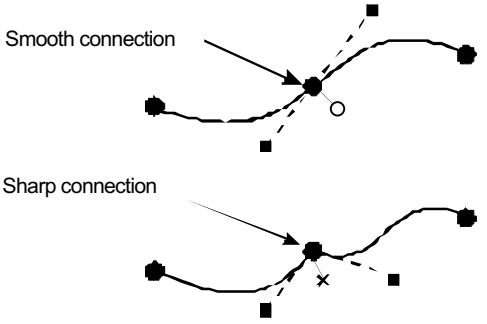
Some TrueType curves may appear linked together and form a long "*supercurve*" where intermediate on-curve points do not really exist, but are implied by the rasterizer:



TrueType curves end with points that look exactly like vector points. Off-curve points of the TrueType curves have a "**plus**" (in black-white mode) or **light-blue** (in color mode) appearance.

Connections

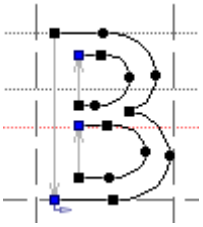
The type of connection between graphical primitives is very important if you want to keep the contour smooth at appropriate nodes. There are two types of connections: sharp and smooth. At a sharp connection, the two connected graphical primitives (curve and curve or vector and curve) are absolutely free in their angle relative to each other at the connecting node. At a smooth connection, the direction of the vector and the control vector of a curve or the control vectors of two sequential curves are kept collinear (lie on the same straight line). I.e. the angle between the two primitives at the node is fixed at 180 degrees.



It is very important to maintain the smoothness of the glyph's contours at the appropriate places. Small corners (sharp connections that are invisible when glyphs are small) become visible (and ugly) when you print large text. Furthermore, rasterizing programs that convert outline glyphs into bitmap images on paper do not like outlines where sharp connections are present in places where the outline should be smooth.

Outline Appearance

You can view an outline in contoured or filled mode. These modes are equivalent for all editing operations, but the filled mode is a little bit slower. However, in the filled mode you always see how the glyph will look in the resulting font. Switch between modes with the button on the **Show Layers** toolbar.



Outline mode

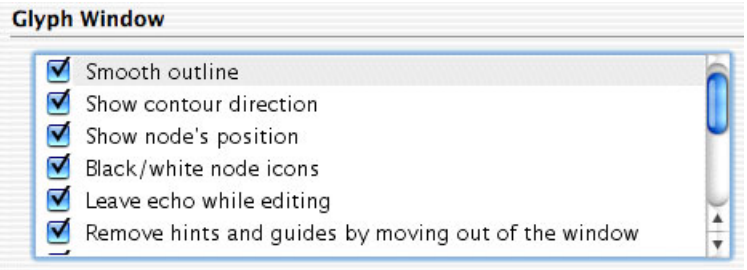
Preview mode

Smoothed Contour

By default contour is rendered with black color and sometimes it may result in jaggies:

Optionally you can smooth the contour appearance on screen, which will result in a much smoother outline appearance:

To smooth contours, use the Glyph Window page of the Preferences dialog box (**Application/Preferences** menu command):



You can use the **Apply** button in the bottom-right of the Preferences dialog box to check the result of the changes you make in Glyph Window options.

➲ **Note:** If your computer is slow and a contour is complex, smoothing the outlines may degrade the performance of the editing tools. Turn it off in this case.

High-quality Preview

No matter which mode is active you can quickly view a high-quality preview of the outline by pressing the ‘`’ key on the keyboard. Until you release the key you will see a high-quality preview of the outline. Note that you can use the ‘<’ and ‘>’ keys to browse characters without releasing the ‘`’ key.

Outline Preview Options

You can choose various options for previewing the contour layer. Use these buttons in the Show Layers toolbar:

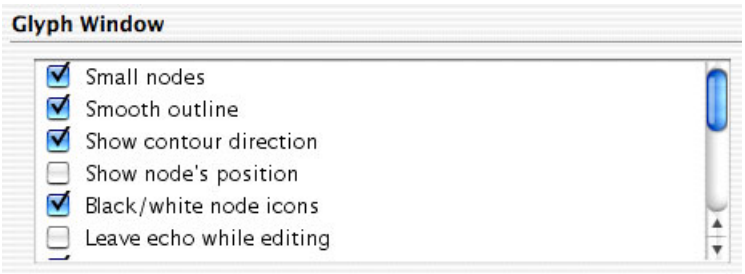
	Nodes	To show nodes or not.
	Control vectors	To show curve control vectors or not.
	Connection mode	To show connection mode marks.
	Positions	To show the coordinates of each node

You will see the positions of nodes only when the nodes are visible. All other contour presentation modes may be combined in any way.

A few more notes about outline appearance:

Selected parts of an outline appear red in color. Selected nodes are marked as red rectangles and they are visible even if non-selected nodes are hidden.

Many options related to outline appearance can be customized on the Glyph Window page of the Preferences dialog box (Application/Preferences):



Here is a list of the available options and a description of the features they control:

Small nodes

Nodes may be small or large:

Smooth outline

Allows one to select between standard and smoothed rendering of the outline:

Show contours direction

An outline consists of several contours and each contour is directional. The direction of the contour is marked with a small arrow:

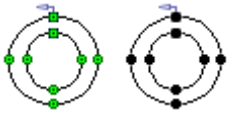
Show node's position

One node may be selected as the current node. It will be highlighted and its position will appear on screen:

To deselect the node, click anywhere in the empty space of the editing field or click the **Esc** key.

Black/white node icons

Nodes may be colored or black/white:

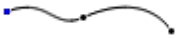


Highlight the first node of the open contour

When this option is on, start and end nodes of the open contour are highlighted with a small diagonal cross:

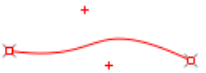
Do not fill open contours

When this option is on, the open contour appears unfilled in preview mode:



Bezier control points are visible in selection

When this option is on and **Show Layers/Control Vectors** is off, the control points become visible when the curve is selected:

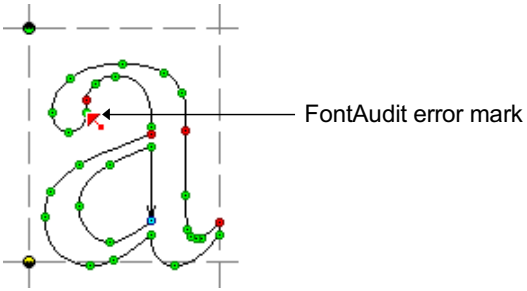


FontAudit

FontAudit is a set of algorithms that analyses a glyph's outline to find errors that may decrease glyph rasterization quality.

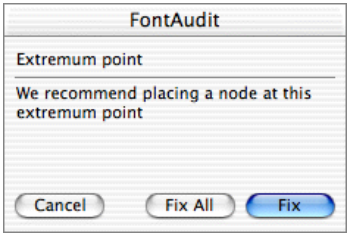
To switch on FontAudit, press the button on the Show Layers toolbar.

If FontAudit finds what it thinks is an error, it shows an error mark in the editing field of the Glyph Window:



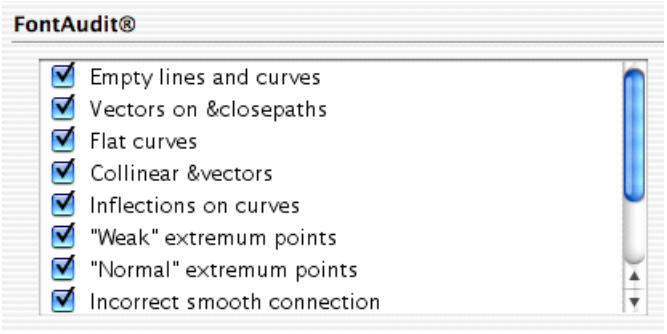
To see a description of the error, activate the Edit tool ( ); position the mouse cursor on the mark; and click the mouse button.

You will see a FontAudit error message:

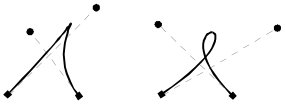
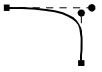


This message has two buttons, **Fix** and **Fix All,** which you can use to try to automatically fix an error (**Fix** button) or all errors in the glyph (**Fix All**). Sometimes correcting one error causes others and when you press the **Fix** (or **Fix All**) button you will see even more errors, so use this feature carefully.

You can customize the FontAudit algorithms using the FontAudit page of the Preferences dialog box:



Here is a short description of each test and the error that it detects:

Empty lines and curves	Lines or curves that have no length. (I.e. two nodes on top of each other.)
Flat curves	Curves that can be replaced with a straight vector without loss of quality. (I.e. a "curve" that is really a straight line.)
Vectors on closepaths	Unnecessary vectors that should be removed. In Type 1 fonts this error can cause problems with rasterization
Collinear vectors	Two sequential vectors are collinear; therefore the first vector can be removed. (Straight lines with extra nodes in the middle.)
Semi-horizontal and vertical vectors	The direction of the vector is close to vertical or horizontal but is not parallel to one of the axes (i.e. not *exactly* horizontal or vertical)
Inflections on the curves	Detects curves that have inflections. It is better to replace such curves with a combination of two curves *Curve with an inflection*
Cusp and self-intersecting curves	 *Cusp curve* *Self-intersecting curve*
Normal extreme points	Curves need nodes at extreme points.
Weak extreme points	There are "invisible" extreme points on curves. This error can cause problems with rasterization of the glyph  *Curve with an invisible extreme point*
Incorrect smooth connection	A vector and curve or two curves are connected very close to a smooth connection, but not precisely. I.e. what looks like it should be a smooth connection is labeled as a sharp connection.
Contour is not closed	Contour appears to be closed (visually) but is defined as open. Use Fix button on the error reporting dialog box to automatically correct this situation.

You can switch off any of the FontAudit tests by switching off their check box in the FontAudit options page.

As you can see, FontAudit testing can detect errors that are invisible without your having to do a tough, lengthy analysis.

Moving Nodes

The most important editing operation is the modification of the contours that build each glyph. You can modify contours in three ways: moving nodes, editing graphical primitives using non-node editing, and selecting several nodes and moving them together.

To move individual nodes:

1. If nodes are hidden, make the node that you want to edit visible: switch nodes on with the **View/Show Layers/Nodes** menu or click near the node to make it visible. If you missed and an incorrect node is highlighted, use the **Page Down** and **Page Up** keys to move to the correct node:

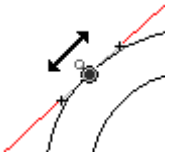
2. Drag the node to the new place. It will stick to other layers if they are visible and snap-to those layers that are activated (**View/Snap-to-layers** menu).

   Hold down the **Shift** key to constrain the direction of the node's movement in 15-degree increments and to snap the cursor to the original node's position.

Options

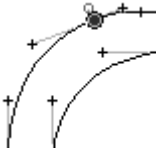
If you are moving a node that is connecting two Bezier curves you have the following options:

1. If the connection of the curves is smooth, press the **Shift** key before clicking the node to constrain movement to a line between the curves' control points:

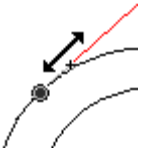


2. If the connection is sharp, press the **Option** key at any time while dragging the node to move it without the adjacent control points:

3. If the connection is smooth, press the **Option** key before moving the connecting node to keep the connection's curvature optimised. Hold down the **Cmd** key to involve all 4 control points in the process:



4. When you are editing control points of a Bezier curve press the **Shift** key before clicking the button to keep the direction of the control vector unchanged.



5. If you are moving a control point of a curve with a sharp connection, press the **Option** key to temporarily change the connection type to smooth, so that the adjacent control vector will be collinear.

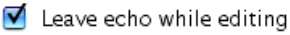
Do not forget that you can press the ‘`’ key at any time to get an instant high-quality preview of the glyph outline as it will print:

Normal outline

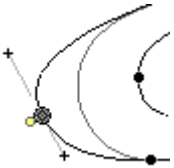
High-quality preview

Outline Echo

If you want to see how the outline looked before you moved a node, switch on the Echo mode. Open the **Glyph Window page** of the Preferences dialog box and switch on this option:



This is how editing field will look when echo mode is on:



Using the Keyboard

You can use the keyboard to move nodes and to select a node for editing:

Arrow keys	Every click moves current node by one font unit.
Shift+Arrow keys	Every click moves a node by 10 font units.
Cmd+Arrow keys	Every click moves a contour that contains a node by one font unit
Cmd+Shift+Arrow keys	Every click moves a contour by 10 font units.
Page Up	Selects the previous node for editing
Page Down	Selects the next node for editing
Tab	Alternates between the node and Bezier control vectors
Esc	Drops the selection of the current node

➲ **Note:** You can make a line or a curve a current object and arrow operations will move it as a whole. Just click on a curve or a line with the Edit tool and it will be highlighted by a pair of short lines:

Non-node editing

Sometimes you may want to modify a contour in a more flexible way than by moving nodes. For example, to adjust the shape of a curve in node editing you would usually make the control points of a curve visible and move them to modify the curve. A more intuitive way would be to “grab” the curve somewhere between the nodes and move this imaginary “inside” point. The curve’s shape changes accordingly. We call this method “*non-node editing*”. This means that you can move not only nodes, but every point of a glyph’s contour. You can even switch off nodes and still be able to edit the contour as you wish.

To modify a curve or vector using the non-node editing method:

1. Move the mouse cursor onto the place on the curve (vector) that you want to move.
2. Press the mouse button. You will see a small color point that will show you the temporary point that you are moving.
3. Drag the mouse and observe how the shape of the curve changes. After a few experiments (which can be undone) you will have enough experience to use this method of editing.

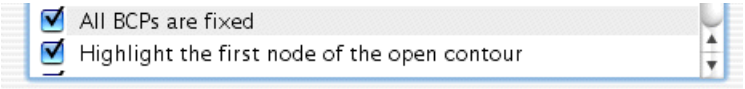
Several notes that you should remember:

1. In non-node editing, guiding objects are not sticky. So, temporary points do not snap to the grid, guidelines, hints or anything else.
2. If you choose a temporary point near one of the ends of a curve, you will move that end, not just change the curve's shape. This is a useful method to locate a curve's endpoints.
3. When you press the mouse button to begin non-node editing you will see that the endpoints of the curve as well as the control vectors appear, simplifying the editing of this primitive.
4. If you want to highlight the line or curve but don't want to modify it, hold down the **CMD** key while clicking on the line or curve.

If you drag a "point" on a curve, its control vectors may change direction:

To fix the direction of the control vectors, hold down the **OPTION** key and double-click the node. You will see the connection mark turn yellow. **OPTION**+double-click it again to remove the fixed state. Alternatively you can **CTRL**-click the node and use the **Connection/Fixed** option to control this feature.

You can also fix the direction of all control vectors when you edit a curve using non-nodes editing. Open the Preferences dialog box (Application/Preferences menu), select the Glyph Window page and use the following option:



Changing Connection Type

The type of connection between graphical primitives is very important in maintaining the smoothness of contours. Connections can be of two types: smooth and sharp.

To make connection marks visible click on the button on the Show Layers toolbar.

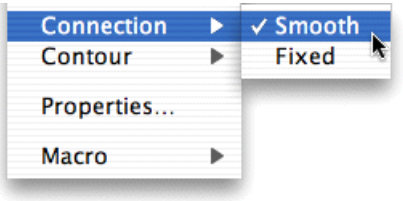
If a connection is smooth the direction of the adjacent curve control vectors or of the curve control vector and line is collinear and the contour is smooth at the connection.

To change the type of connection

1. Make the node visible.

2.1 Double-click the node

2.2 CTRL-click the node and select the connection type in the popup menu:



Deleting Nodes

To delete nodes using the Edit tool:

1. Begin moving the node by dragging it.
2. While dragging, **Ctrl**-click the mouse button. The node will be removed.

---

1. Move the mouse cursor onto the node and **Ctrl**-click the mouse button.
2. In the popup menu choose the **Delete node** command.

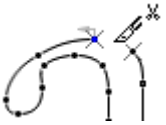
➲ **Note:** If you **Ctrl**-click the mouse button while editing the curve using the non-node editing method or while you are moving the control points of a curve the curve will not be removed. Instead it will change to a straight line.

You can use the **Edit/Delete** and **Edit/Cut** commands to remove nodes and other selected elements of a glyph as well.

Usually, deleting a selection breaks the contours. To keep them closed instead uncheck the **Edit/Delete command breaks contour** checkbox in the Glyph Window page of the Preferences dialog box.

Deleting Lines and Curves

You can delete a whole line or curve with the Knife tool. Activate the Knife tool with the button on the Edit toolbar or click '3' on the keyboard. Press the **Option** key and click on the line you want to delete. Note that with this method you will always break the outline:

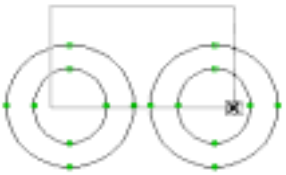


This method breaks the contour even if the **Delete command breaks contour** checkbox in the Glyph Window page of the Preferences dialog box is unchecked.

You can use the **Edit/Delete** (the **Del** or **Backspace** key) and **Edit/Cut** commands to remove the active curve or vector as well.

Eraser Tool

The eraser tool can be used to quickly remove nodes. Sometimes this is necessary, for example, with contours from an auto-tracing program. The eraser tool can work in two modes: like a standard eraser or as a rectangle eraser. In the first mode, all nodes that are inside the rectangle of the eraser mouse cursor are deleted. In the second mode, you define a rectangle by clicking and dragging (as when you select nodes with the **Edit** tool or change the zoom of a Glyph Window) and all the nodes inside the rectangle are removed.



The first (eraser-like) mode is the default for the **Eraser** tool. To switch to the rectangle mode, hold down the **CMD** key.

Inserting Nodes

To insert a new node on a vector or curve with the Edit tool:

1. Press and hold the **Cmd** and **Option** keys.
2. Position the mouse cursor on the point where you want to insert the node and click the mouse button. The mouse cursor will turn into the knife and the new node will appear in that place.

Using the "position" tool to insert nodes:

1. Activate the so-called position tool by pressing the mouse button **on a vector or curve** while the **Ctrl** and **Option** keys are also pressed down. The mouse cursor will turn into a "target" with its coordinates shown.
2. Drag the mouse cursor along the primitive (it will stick) to "take aim" precisely and release the mouse cursor where you want the node to be inserted.

Using the Knife tool to insert nodes:

1. Activate the Knife tool .

2.1. Click on the point on the contour where you want to insert a node.

2.2. Press the mouse button anywhere on the empty area of the editing field and drag the mouse to form a "knife line". After you release the mouse button new nodes will be inserted at all points where this line crossed the outline. Hold down the **Shift** key to constrain the direction of the "knife line" to 15-degree increments.

If the "knife" line will cross two lines you may find a part of the glyph to "cut off". Hold the **Option** key to limit Knife tool to insert new nodes only.

Using the Add Corner, Add Curve and Add Tangent tools:

1. Activate one of the tools.
2. Click on any outline point. The Corner tool will add a straight line; the Curve tool will add a smooth connection and curve and the Tangent tool – a sharp connection and curve.

➲ **Note:** You can insert nodes on the closing vector that automatically connects the first and last nodes of a contour. If you insert nodes on the first half of a closing vector (closer to the ending node of a contour), then the new node will be added to the contour. If you insert the node on the last half of the closing vector, then it will be inserted before the starting point.

Using the Drawing Tool

The easiest way to create a new contour is to use the Drawing tool: .

You can create a new contour or you can continue any existing contour. If you want to add new nodes to the existing contour, just activate its first or next node:

The last node of the open contour is activated

1. **To add a point**, just click the mouse button.

2. If you want **to create a line point**, just release the button. If you want **to define a curve**, drag the mouse to set the position of the curve control vector:

3. **To adjust the position of the curve control vector** without moving the control vector of the previous curve, hold **Option** and drag:

   You can press and release the **Option** key while you drag the mouse – when **Option** is released you are defining the positions of the control vector that belongs to the previous curve *and* the control vector of the next curve. When **Option** is pressed, you are not moving the previous curve's control vector.

4. When you are adding a new node, you can hold the **Cmd** key to not move the curve control vector but **move the node** itself.

5. Finally, you can use the **Shift** key at any time **to constrain the direction** of the line (if you are holding the **Cmd** key) or a curve control vector.

To close the contour, just click on its starting point and drag the mouse to set the direction of the control vectors.

Adding Points to a Contour

In addition to the Drawing tool you can use three more tools to create a new contour or to add points to an existing contour. These tools are: Add Corner, Add Curve and Add Tangent.

To create a new contour:

1. Activate one of the tools.
2. Click anywhere in the empty area of the glyph window to create the first point of a new contour. Drag the mouse to put new node into correct position.
3. Click again in the empty area to add a corner line, curve or smoothly connected curve (with Add Corner, Add Curve or Add Tangent tools respectively).
4. Continue the procedure until your newly defined contour is complete.
5. You can switch to the Drawing tool at any time and use it to add new points to a contour you are creating.

➲ **Note:** A new node is added to the contour if the last node of the contour is highlighted. If it is not highlighted a new contour is started.

To highlight a node click it. To deselect it, press the **Esc** key on the keyboard.

You can move outline nodes with the Add ... tools. Note that if you click on the contour (not on the node), a new node is inserted. The type of node depends on the tool you are using. To prevent adding a new node hold down the **Option** key when you click on the contour.

Converting Primitives

Sometimes you may want to convert a curve to a vector or vice versa**. To convert a curve to a vector** "delete" (**Ctrl**-click while dragging) one of the control points of the curve, or "delete" the curve while you are in the non-node editing mode.

To convert a vector (normal or closing) to a curve drag an inside point of the vector while holding down the **Option** key or drag one of the nodes toward the moddle of the vector holding down the **Option** key.

To convert a curve to a 1/4 part of an ellipse (the curve's control vectors will be treated as an ellipse axis), press the **Option** key and click on the curve.

You can also **convert** curves and vectors **with the popup menu**. **Ctrl**-click the end node of the segment and select the **Convert** command in the popup menu. With this command a line segment is converted to a Bezier curve, a Bezier curve to a TrueType curve and a TrueType curve to one or more Bezier curves.

The last way to convert primitives is to **use the selection menu**:

1. **Shift**-click any point on the curve or line segment.
2. **Ctrl**-click the highlighted (red) segment. A popup menu appears.
3. Choose one of the commands in the **Convert** submenu: **To curves** (to convert to a Bezier curve) or **To lines** (to convert to straight line segments).

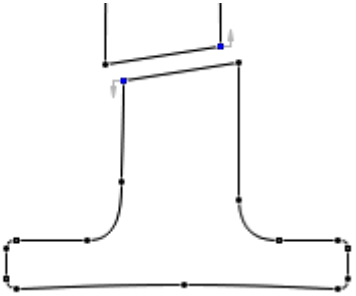
Breaking and Joining Outlines

To break the contour with the edit tool hold down **Cmd** and **Option** and click the node where you want to break the contour.

To break the contour with the Knife tool just click on the node.

When a contour is broken its first and last nodes are highlighted by diagonal crosses:

You can use the Knife tool **to "cut out" part of the outline:**



1. Activate the Knife tool.
2. Press the mouse button and drag the cursor to define the "cutting line".
3. Release the button. Note that you can only cut part of a single contour, like in the sample picture above.

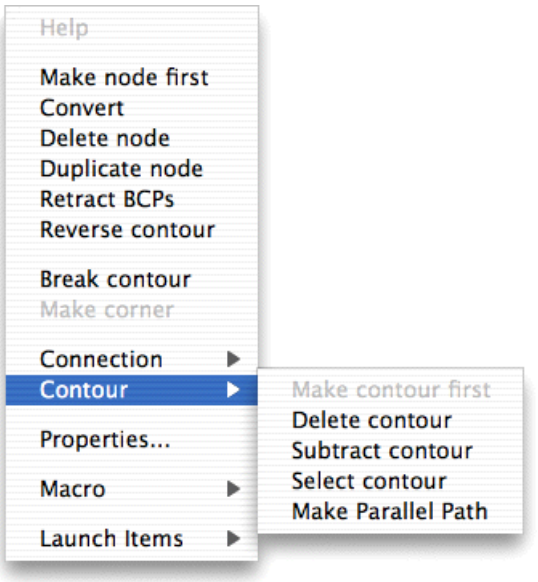
➲ **Note**: Deleting a selection with the **Edit/Delete** command breaks contours too. To keep them closed uncheck the **Edit/Delete command breaks contour** checkbox in the Glyph Window page of the Preferences dialog box.

To join two contours you need to move the starting or ending node of one contour to the starting or ending node of another contour.

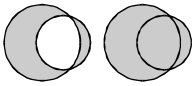
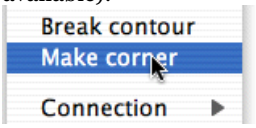
Hold down the **Option** key **to prevent the contours from joining**.

Node Commands

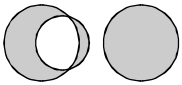
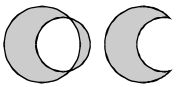
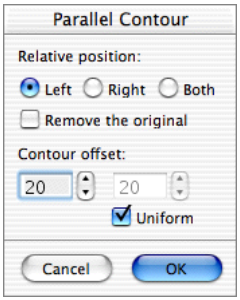
If you **Ctrl**-click a node you will see a popup menu with many useful commands:



Below is a description of all the commands in this menu:

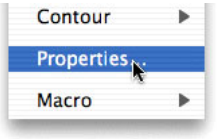
Make node first	Starts the current contour from the selected node (i.e. makes it the startpoint). This command is useful when you need to join contours since you can only connect starting and finishing nodes.
Convert	Cycles the node type from line to Bezier curve to TrueType curve.
Delete node	Removes the node.
Duplicate node	Adds a zero-length vector to the node. I.e. puts a new node on top of an old one. This command is useful when you are editing Multiple Master fonts.
Retract BCPs	Removes the control vectors of the node, making it sharp.
Reverse contour	Reverses the contour direction: 
Break contour	Breaks the contour at the selected node.
Make corner	Makes a 90 degree corner (this operation is not always available): 
Connection	Popup menu with connection settings. You can use it instead of double-click or Option+double-click.
Contour	Set of commands related to the contour to which the selected node belongs (described below).
Properties	Opens the Node properties panel.

Contour commands:

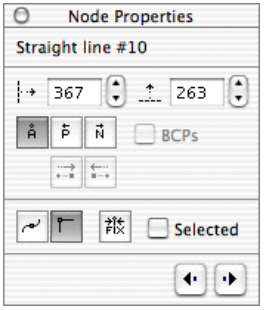
Make contour first	Begins the glyph from the selected contour. Renumbering contours is important for hint-substitution programming.
Delete contour	Removes the contour 
Subtract contour	"Subtracts contour" from the outline 
Select contour	Reverses the selection state of the contour.
Make parallel path	Creates a contour that is parallel to the current one. After you select a command you will see a dialog box with the path options:  Select the options (relative position of the path and the offset) and click **OK** to generate new path. Check the **Remove the original** check box to remove the contour you clicked and keep only the generated parallel contour.

Node Properties

Cmd-click the node or **Ctrl**-click and choose the **Properties** command in the menu:

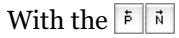


You will see the Node properties panel:



In this property panel you can control the position of the node, the alignment type, the selection status of the node and the position of the control points of the curves.

To change the position of a node:

1. Select the origin point you want to use and set the coordinates of the node. By default the origin is the glyph's origin point . With the  buttons you can select the previous or next node as the origin point.
2. Modify the coordinates of the node in the edit boxes. You can use the spin buttons to increase or decrease the coordinates. The new coordinates will be applied to the node when you press the **ENTER** key on the keyboard or move the focus from one edit control to another or when you close the property panel by clicking on a free space in the edit field.

To change the connection mode for a node: select the new mode by clicking on of the buttons in the lower part of the panel.

To change the selection state of a node: modify the state in the **Selected** check box.

To edit the position of the curve's control vectors: switch on the **BCPs** check box (it will be gray if you are editing a node between two vectors) and modify the relative position of the previous or next control point that belongs to that node.

Use the buttons with arrows **to edit the previous or next node**.

➲ **Tip:** when you are editing node positions in the Properties panel, press the **ENTER** key to accept changes and move the focus to the editing field of the Glyph Window. There you can use the keyboard to move the selected node and the **PAGE DOWN**/**PAGE UP** keys to select another node for modification. You will see the node properties change in the Properties panel as you move the node by keyboard or mouse. Click **OPTION**+**ENTER** to put the focus on the Properties panel to set the node position more precisely.

Previewing Glyphs

Sometimes you need to get a high-quality preview of the glyphs you're editing. You may preview any single glyph with the '`' key, but it doesn't give you a feeling of how this glyph will look in a line of text or in multiple sizes.

To get a better preview, use the Preview panel. Open it with the **Preview panel** command in the **Window** menu or by clicking the button on the Panels toolbar.

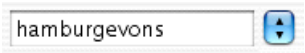
You will see a panel containing three pages: OpenType Features, Preview and Anchors. The OpenType Features and Anchors pages will be described later, so activate the Preview panel now:

The Preview panel can work in two modes: string preview and waterfall preview. You can choose the mode with this button: . Note that the waterfall preview is possible only when the Glyph Window is active.

You can **scroll the sample string** in the Preview panel – just drag to scroll the window or **Ctrl**-click to reset to the initial position.

High-quality rendering provided by the FreeType library is used to show glyphs in the preview panel, so the visual resolution is higher than the pixel resolution of your screen and is close to what you can get on some printers.

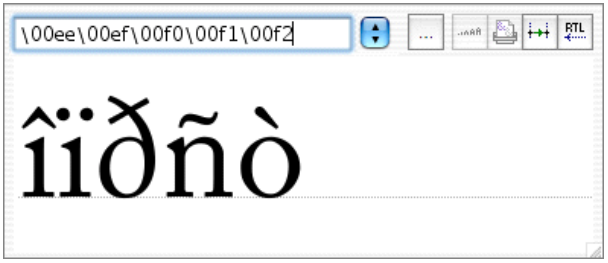
In preview mode the preview panel shows the glyph string in the top area of the window:



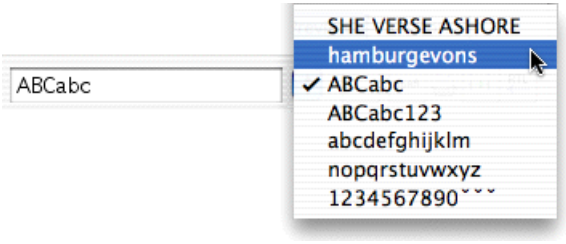
You can type any text in this area or you can **drag-drop glyphs selected in the Font Window or in the Collection/Classes panel**.

To preview a glyph that cannot be easily encoded by a single keyboard character, enter the slash character and then the name of the glyph:

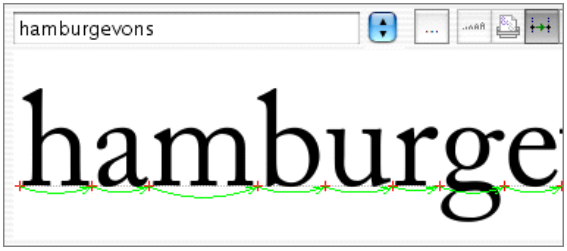
To enter Unicode characters, type the Unicode index (in hexadecimal form, exactly 4 characters) after the backslash:



To choose one of the predefined sample strings, open the list with the popup button to the right of the sample string:



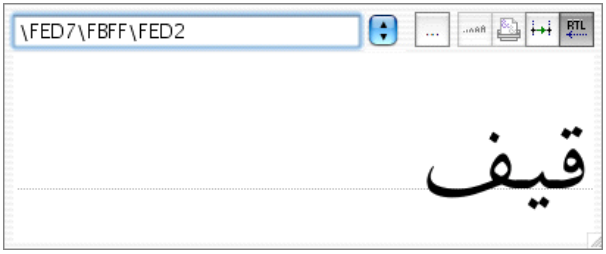
With this button you can activate metrics preview mode:



In this mode you can see a preview of every glyph and its width.

The height of the sample string is determined automatically and depends on the vertical size of the preview panel.

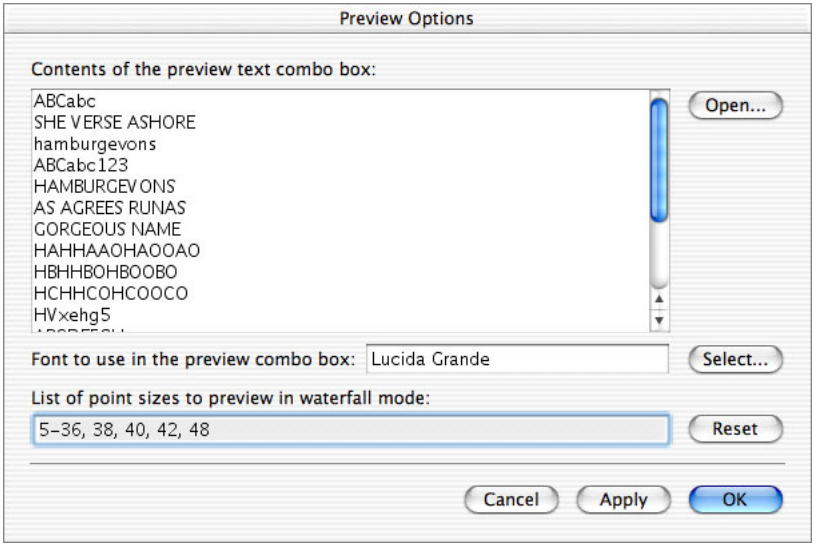
If you are working with Arabic or Hebrew script you may need to preview a sample string in right-to left mode. Use this button to activate it:



Preview Options

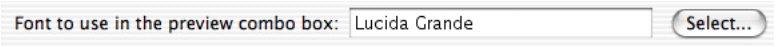
To edit the contents of the sample strings list, click on this button:

You will see a dialog box that contains all the strings that appear in the list plus a few other options:

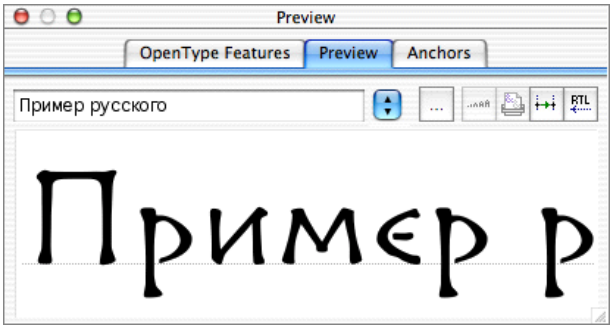


By clicking the **Open** button you can select the text file that will be used to preview sample strings.

This option:



allows you to choose the font that is used to show text in the sample string. You may need to change this font if you are working with non-Latin fonts:



Vertical Preview

If the width of the preview panel is less than half of its height (which happens if you dock the preview panel to the right or left side of the screen), it automatically turns itself into vertical preview mode:

A font's vertical metrics or glyph's vertical metrics (if defined) are used to vertically align glyphs. Right-to-left preview mode still works when the Preview panel is vertically oriented, so you can align glyphs to the left or to the right to check the balance of their sidebearings.

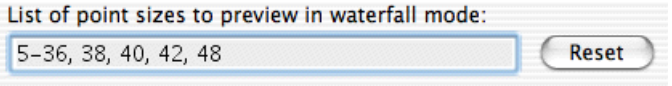
Waterfall Preview

When you are working on a glyph, you may need to see it in many sizes at once. Open the glyph in the Glyph Window and switch the Preview panel to the waterfall mode with this button:

You will see multiple sizes of the glyph previewed:

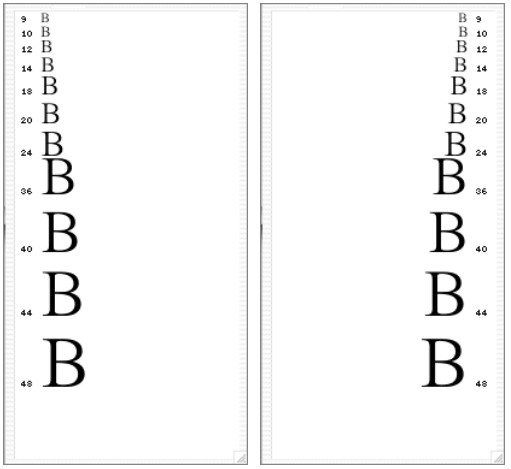
Size is measured as PPM, which means Pixel Per eM. This is a resolution-independent way of defining the pixel height of the glyphs. For standard Mac screens, the point size (which you enter to define text size in text editors or page layout programs) is equal to the PPM size.

With the Options dialog box (...) you can choose which PPMs you want to preview:



You can choose what sizes you need to see in the waterfall preview mode. Enter the sizes separated by ‘,’ or use ‘-’ to define size ranges. Click the **Reset** button to select the default PPM ranges.

Waterfall preview works in vertical mode with left-to-right or right-to-left writing mode:



You may print the Waterfall preview of the font if you click on the button on the preview panel toolbar.

Selections

Many operations can be applied not only to single nodes or graphical primitives but also to several nodes together. For example, you may want to move many nodes or delete part of a contour. First, select the nodes that you want to process.

To select nodes with the selection rectangle:

1. Make sure that the Edit tool is active.
2. Press the mouse button anywhere in the empty area and drag the mouse to surround the nodes with a rectangle. Hold down the **Shift** key to reverse the selection state of the nodes.

To select or deselect individual nodes just shift-click them.

To select the contour segment (line or curve) – shift click on it.

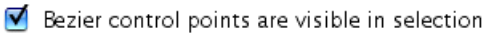
To select the whole contour double click the contour (not a node) with the mouse cursor. To add another contour to the selection **Shift**-double click it. Alternatively you may press the **Cmd** key and click anywhere on the empty area close to the contour (the Magic Wand cursor appears). Hold down the **Cmd** and **Option** keys to select nested contours too.

To select all the contours in a glyph use the **Edit/Select all** command.

To deselect all nodes click the mouse button somewhere in the free space of the editing field or use the **Edit/Deselect** command.

To reverse the selection state of all nodes in the glyph use the **Edit/Invert selection** command.

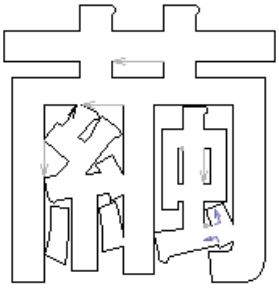
If the option in the Glyph Window page of the Preferences dialog box



is on, then all the control points of the selected nodes will be visible even if the View/Show Layers/Control Vectors option is off.

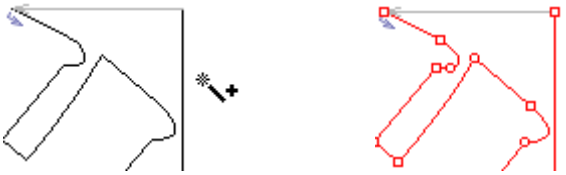
Using the Magic Wand Tool

With the Magic Wand tool you can easily and precisely select contours. It is especially useful when you are working with glyphs that have many contours, such as Far-Eastern ideographs:

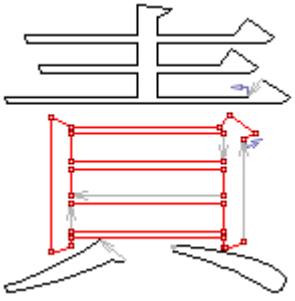


To select the contour with the Magic Wand tool just activate the tool and click anywhere near the contour. You don't need to be precise – FontLab will automatically locate the closest contour.

To reverse the selection state of the contour, hold down the **Shift** key and click anywhere near it:



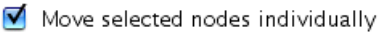
You can also select a contour and all contours that are inside it. Just hold the **Option** key when you using the Magic Wand tool:



Moving the Selection

You can move the selected part of the contour by mouse – drag any selected part of the contour or use the arrow keys. If you press the arrow key then the selection will move in that direction by one font unit. Hold down **Shift** while pressing the arrow keys to accelerate the movement of the selection.

If the option in the Glyph Window page of the Preferences dialog box

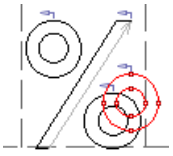


is on, then dragging node won't move the whole selection but only this node.

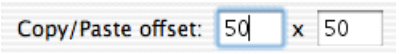
Copying the Selection

Sometimes you need to copy glyphs or parts of glyphs to another place in the font or even into a different font. With FontLab you can put any part of a glyph or an entire glyph (with hints, guides, etc.) into the Clipboard and paste it into a different place.

To copy parts of the glyph's outline use the commands from the **Edit** menu or the buttons on the Standard toolbar:

Cut		To copy a selected part of the glyph onto the Clipboard and delete it from the glyph.
Copy		To copy a selected part of the glyph onto the Clipboard.
Paste		To add a contour part copied to the Clipboard into the current glyph as a new contour
Insert		To replace the current selection with the Clipboard contents.
Delete		To remove the selected part of a glyph's contour.
Duplicate		To insert a copy of the selection into the current glyph as a new contour 

When you use the **Paste** command, the selection is offset from the original location. You can change the amount of this shift in the Preferences/Glyph Window dialog box (**Application/Preferences** command):



➲ **Note**: When you paste the glyph outline copied in the Font window, it is always pasted without a shift.

On the same page you can adjust the amount of shift during the Duplicate operation:

Because the Clipboard is used as a buffer for copying contours you can paste glyphs and their parts not only to the current font but to any glyph of any font of any application that is compatible with FontLab (ScanFont 3.0, for example).

Copying an Outline to Other Applications

With FontLab you can exchange outline data with other vector-editing programs. Supported programs are Adobe Illustrator and Macromedia FreeHand.

To copy part of the glyph's outline to a vector-editing program use the usual copy procedure. The selected portion of the outline will be copied to the Clipboard. Then switch to your vector-editing program and select the **Paste** command from the **Edit** menu.

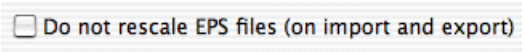
You can change the size of the copied outline using the scale transformation that is supported by your vector-editing application.

To copy an outline from a vector-editing program into FontLab select the outline object that you want to copy and choose the **Copy** command from the **Edit** menu (in the source application). To place the copied outline in FontLab switch to FontLab (Glyph Window) and select the **Paste** command from the **Edit** menu.

Notes:

1. FontLab can only edit font outlines, not features such as color of outline, outline width or fill color. Regardless of the settings you have in the vector-editing application, only information about outlines will be copied to FontLab.
2. To see how an outline will look in FontLab before you copy it, in the source vector-editing application select the object and set the fill color to 100% black, and the width of the outline to none.

When you paste an outline from another application into FontLab it scales it to fit the font's UPM. To prevent the outline from automatically scaling, use the **Do not rescale EPS files** option located at the top of the **Preferences/General** dialog box:



Transform Selection

Sometimes you need to scale, rotate or slant a whole or part of a glyph outline. In FontLab you can do this using different methods:

1. Using the Transformation panel (described below)
2. Using the Transform tools
3. Using the Transform Selection operation

Using Transform Tools

In Edit mode you have access to three transform tools:

	Rotate	Rotates the contour
	Scale	Scales the contour
	Slant	Slants the contour vertically or horizontally.

To transform the outline:

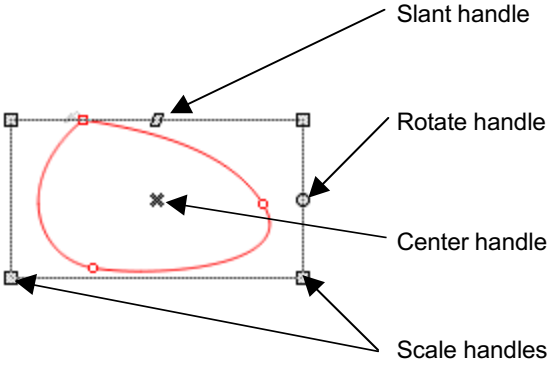
1. Select part of the outline you want to transform or undo all selections to transform the entire glyph outline.
2. Activate one of the transform tools.
3. Position the mouse cursor at the center of transformation, press the mouse button and drag the mouse to make the transformation. Remember that you can press the ‘`’ key at any time to get a high-quality preview of the transformed outline.
4. Use the **Shift** and **Cmd-Shift** keys to constrain the transformation.
5. Release the mouse button to complete the transformation of the outline.

Using the Transform Selection Operation

To activate the Transform operation click the **Transform Selection** button in the Tools toolbar or **Ctrl**-click the selected portion of a glyph and select **Transform** in the popup menu.

Or you can double-click on any selected graphical primitive to activate the **Transform** operation.

When this operation is activated, you will see a transformation rectangle surrounding the selected area. If nothing is selected, the entire glyph will be subject to transformation.



So, what do all these handles mean, and how can they be used?

To move a selection:

1. Position the mouse cursor somewhere inside the transformation rectangle but not on the center handle.
2. Press the mouse button and drag the rectangle to its new place.
3. Release the button. The selection will be moved.

To scale or skew a selection:

1. Position the mouse cursor on one of the scale handles ▫.
2. Press the button and drag the mouse. You will see that the transformation rectangle is scaled. Hold down the **Shift** key on the keyboard to constrain the scale proportionally.
3. Release the button when you are done. The selection will be modified.

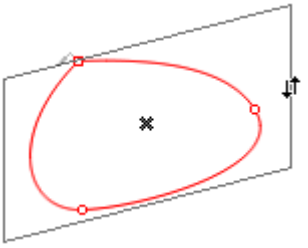
To rotate a selection:

1. Move the mouse cursor onto the rotation handle ○.
2. Press the mouse button and drag the mouse. The transformation rectangle will rotate around its center. Hold down the **Shift** key to constrain the rotation angle to 15-degree increments. You can also use the slant handle for rotation – just press the **Option** key to alternate between rotate or slant.
3. Release the button to accept the rotation.

To move the center of rotation just drag the center handle ✖ by the mouse to its new position.

To slant a selection:

1. Move the mouse cursor onto the slant handle .
2. Press the mouse button and drag the mouse. The transformation rectangle will be slanted. Hold down the **Shift** key to constrain the rotation angle to 15-degree increments.
3. To slant in vertical direction, hold the **Option** key and drag the rotate handle:



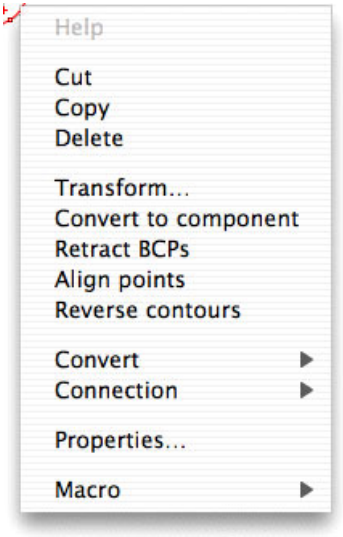
4. Release the button to accept the slanting.

Double-click in the editing field or press **Enter** to accept the completed transformation or press the **Esc** key to reject it.

You can use the arrow keys while the Transform tool is active to move the selection by one font unit in the direction of the key you clicked. **Shift+arrow** keys move the selection by 10 font units at each key click.

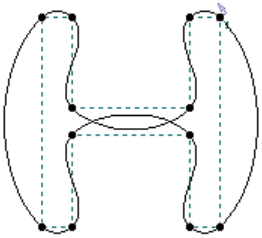
Selection Commands

When a part of the glyph is selected, **Ctrl**-click it to get access to the popup menu:



Some commands are just copies of the **Edit** menu commands, but others are much more interesting:

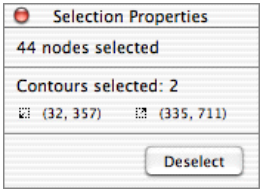
Cut	Copies the selection to the Clipboard and removes it.
Copy	Copies the selection to the Clipboard and leaves original untouched.
Delete	Removes the selection.
Transform	Activates the Transform tool.
Convert to component	Tries to convert the selection into a link to a component glyph. This extremely powerful command is described in full detail in the Editing Composite Glyphs section.
Retract BCPs	Removes the control vectors of the selected points, making them sharp.
Align Points	Aligns the selected nodes vertically or horizontally.

Reverse contours	Reverses the direction of selected contours.
Convert	Converts all selected segments to lines, Bezier curves or off-curve points.
Connection	Changes the connection mode of the selected points. If the mode is changed to smooth, all connections are forced to be smooth: 
Properties	Opens the Selection properties panel.

Selection Properties Panel

To make the Selection Properties panel visible, choose the **Properties** command in the selection context popup menu or use the **Option-Return** keyboard shortcut.

The Selection Properties panel is very simple:

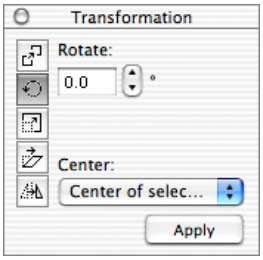


It contains the following information about the selection: the number of selected nodes, the number of selected contours, and the selection bounding box's bottom-left and top-right corner coordinates.

You can click on the **Deselect** button to discard the selection and get the Glyph Properties panel instead.

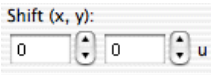
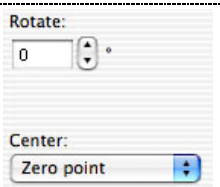
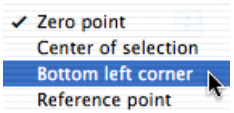
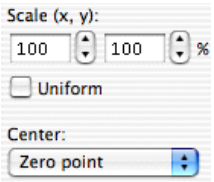
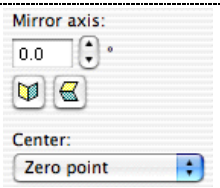
Transformation Panel

The Transformation panel allows you to apply several simple transformations to the selected area or to the whole glyph. To open the Transformation panel you can select the **Transformation Panel** command in the **Window** menu or click on the button on the Panels toolbar:



To transform the glyph or the selected area:

Select the type of the transformation by clicking on one of the buttons in the left and the transformation options in the right area:

Shift		Enter a distance to move the selection
Rotate **Slant**		Enter the rotation or slant angle (degrees, clockwise) and select a center of rotation: 
Scale		Enter the scaling factor and select a center point of transformation. Use the Uniform option to scale proportionally.
Mirror		Enter the direction of the mirror axis and select the center point of the transformation. Use the buttons to mirror horizontally or vertically quickly.

Press the **Apply** button to apply the transformation to the selected area.

Find and Replace Operation

In FontLab you can perform a unique outline find/replace operation that allows you to replace an identical part of an outline that occurs in many glyphs with an alternative form. It is also very useful in analysing an outline for correctness.

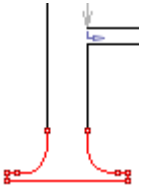
Here is a typical situation where you could use the **Edit/Find** command:

Original glyph

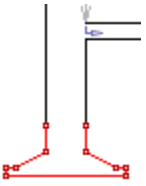
Modification

It is relatively easy to make the change above in a single glyph, but what if you want to do the same thing in all glyphs of the same font that have serifs?

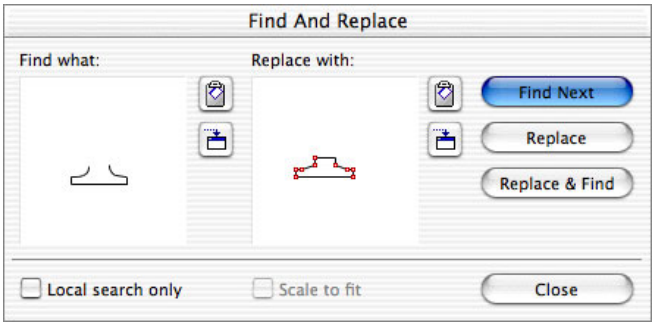
1. Select the element that you want to find:



2. Copy the selection to the Clipboard with the **Edit/Copy** command.
3. Modify the original shape as needed and select the new version of the serif:



4. Using the **Edit/Find** menu command open the Find and Replace dialog box:



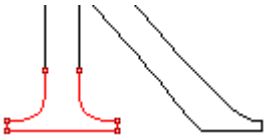
The left panel shows the outline that FontLab will look for and the right pane shows the outline it will use as the replacement.

The two buttons to the right of each pane mean:

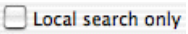
	Place the Clipboard contents into the pane.
	Place the current selection into the pane.

In our sample you need to place the Clipboard contents (original outline) into the left pane (click on the button) and the selection into the right pane ( ) button.

5. Click the **Find Next** button to find the next appearance of the source outline in one of the glyphs. The first such glyph located will appear in the Glyph Window and the outline that is found will be selected:



6. Click **Replace** to replace the old outline with the new one or click **Replace & Find** to replace and find the next location of the outline.

7. Check 

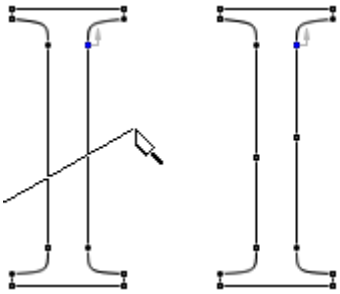
to limit the search area to the current glyph only. If this option is unchecked, FontLab will look for the outline in all glyphs of the font.

Building an Outline From Blocks

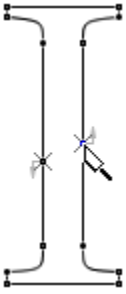
Now you know how to select parts of an outline and copy it, so let's do a few experiments to show how to use this knowledge.

Suppose that we have an 'I' character and we want to create an 'H' character.

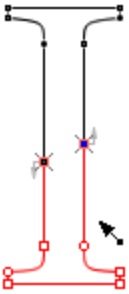
1. Open the 'I' in the Glyph Window (double-click the 'I' cell in the Font Window, use the Glyphs Bar in the Glyph Window or just browse the font with the arrow buttons).
2. Cut the character in the middle. Activate the Knife tool, press the mouse button at the left of the character, hold the **Option** key on the keyboard and drag the mouse cursor to the right to define a cutting line. Release the mouse button:



3. Click on each inserted node to break the contour:



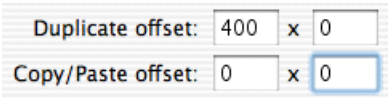
4. Select the bottom half of the 'I'. Temporarily activate the Edit tool ("click" on the **CMD** key on the keyboard) and **CMD**-click near the bottom area of the glyph:



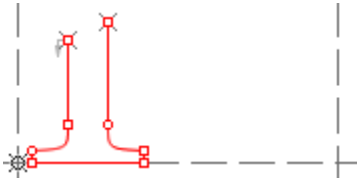
5. Copy it to the Clipboard with the **Edit/Copy** command.

6. Go to the 'H' character. Use the **Glyph/Reset** command to remove the contents of all editing layers.

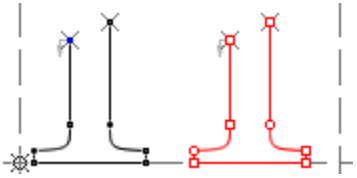
7. Open the **Preferences/Glyph Window** dialog box and set the **Copy/Paste offset** to zero and **Duplicate offset** to 400 x 0:



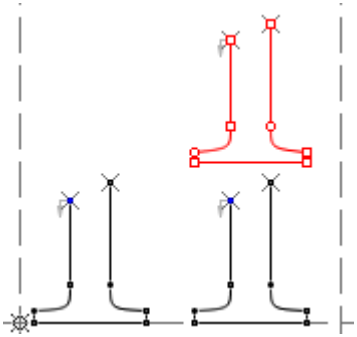
8. Close the Preferences dialog box and use the **Edit/Paste** command to place a copy of the 'I' bottom:



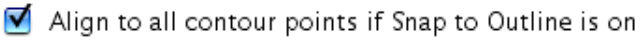
9. Use the **Edit/Duplicate** command to make a second copy:



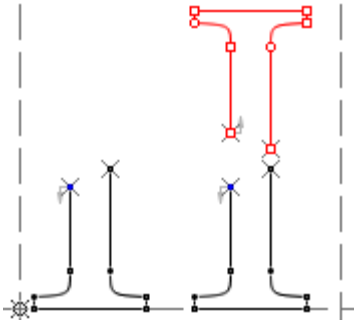
9. Duplicate it again and drag it to a place above the first segment:



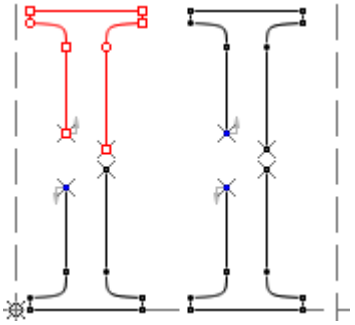
Use the contour snap function to position the segment. Activate the contour snap with the **View/Snap to Layers/Outline**. We also recommend that you activate the feature that will snap a point which you are moving to all outline nodes, by X and Y direction independently. Use the **Application/Preferences/Glyph Window** dialog box:



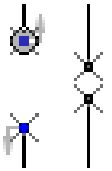
10. Use the **Tools/Outline/V Mirror** command to flip the selected segment:



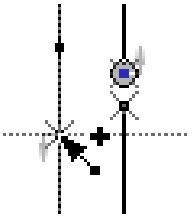
11. Duplicate the top segment and locate the copy above the bottom-right segment.



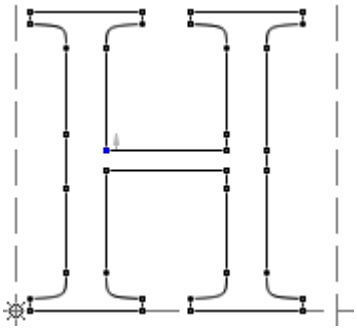
12. Click on an empty area of the Glyph window and then click on the top-left node of the top-left segment to highlight it:



13. Activate the **Add Corner** tool click somewhere and drag the line to connect the left line of the top-left and bottom-left segments:



14. Then click on any other starting or ending node of the contour segments and use the **Add Corner** tool to connect them



You'll notice it takes more time to read the instructions than to actually perform the procedure.

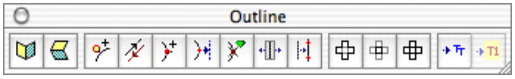
Outline Actions

You can find the actions described below in the **Tools/Outline** menu:

H Mirror	Makes a mirror transformation in the horizontal direction. This operation is applicable to a selection or to a whole outline if nothing is selected.
V Mirror	Makes a mirror transformation in the vertical direction. This operation is applicable to a selection or to a whole outline if nothing is selected.
Check Connections	Analyses an outline and detects the types of connections between outline primitives (lines and curves).
Reverse All Contours	Reverses the direction of all contours of the glyph. This action can be useful if you want to completely convert a Type 1 glyph to a TrueType glyph. In TrueType and Type 1 format the default direction of the contours are opposite.
Nodes at extremes	Automatically inserts nodes at the extreme points of curves. We highly recommended that all curves have nodes at their extreme points.
Snap to Guides	Move all nodes on to guidelines, hints or grid if they are sufficiently close to them. This action will “snap” nodes only to the guiding layers that are currently visible.
Optimize	Optimizes the outline (see the section below).
Expand strokes	Converts contours to brush strokes.
Make Parallel Path	Creates a contour that is parallel to the existing contour. See the next section for more information about this and previous features.

	Merge Contours	Combines all overlapping parts of the outline. This operation and the two following operations are applied to all contours that have at least one node selected. If nothing is selected, they will be applied to the whole glyph.
	Get Intersection	Leaves only areas that are covered by at least two contours.
	Delete Intersection	Removes all areas that are covered by more than one contour.
	To TrueType Curves	Converts all Type 1 (3rd-order) curves to TrueType (2nd-order) curves.
	To Type 1 Curves	Converts all TrueType (2nd-order) curves to Type 1 (3rd-order) curves.

To get quicker access to the commands in **Tools/Outline** menu, select the **View/Toolbars/Outlines** option. It switches on the Outlines toolbar:



Creating Contours

FontLab has two commands that can help you automatically create contours. These commands are **Expand strokes** and **Make parallel path**.

Both commands will take one or more existing contours and create new ones using some options.

Expand Strokes

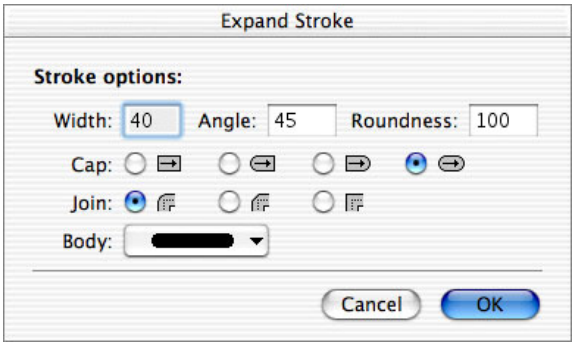
This command will use contours as a trajectory for the paintbrush:

Source open contour

Stroke is expanded with the shaped brush

To use this operation first select the contours to which you want it to apply. All contours that have at least one node selected will be processed. If nothing is selected, the whole glyph contour will be taken.

Select the **Tools/Outline/Expand strokes** command in the menu. You will see a dialog box that lets you select options:



The top area of this dialog box is used to specify brush size and shape. **Width** is the width of the brush ellipse at its widest part. **Angle** is the degree of the brush ellipse slant and **Roundness** is the relation (in percent) of the narrow and wide widths of the brush ellipse. Below is a sample of the stroke expand with different brushes:

The next line of options specifies the way the expanding algorithm will process the contour corners and the ends of an open contour:

Flat contour ends vs. round ends

The last option lets you specify the shape of the brush stroke:

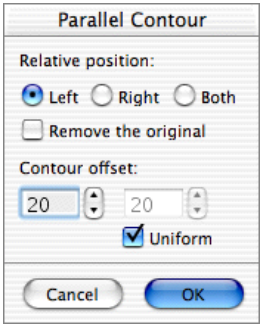
This is the sample of the stroke above with different stroke shapes:

When you finish selecting options, click **OK** to expand the selected contours.

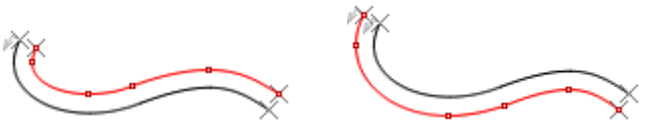
Make Parallel Path

With this operation you can create a path that is parallel to any existing path. The method of selecting contours for processing is simple: all contours that have at least one node selected will be taken into account. If no nodes are selected, the whole glyph outline is processed.

Select the contours that you want to process and choose the **Tools/Outline/Make parallel path** command from the menu. You will see a dialog box for specifying the options of the command:



The relative position section lets you specify the side of the original path at which the new path will be created:



Left and right side option (original path is not selected)

Select the **Both** option to generate parallel contours on each side of the source contour.

Contour offset defines the distance at which the new path will be created. Use the **Uniform** option to have the same distance set for X and Y directions.

Check the **Remove the original** check box to remove the source contours and keep only the generated parallel paths.

Below is a sample of what you can do with the parallel path feature and contour-editing tools:

Glyph contour after multiple parallel paths created and then closed with the Add corner tool

Rendering of the glyphs above

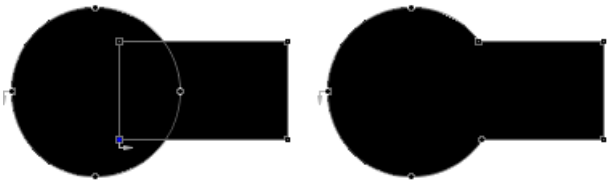
As you can see, with the “Make parallel path” feature you can create complex ornaments in minutes.

Merging and Intersecting Contours

With the **Merge contours**, **Get intersection** and **Delete intersection** commands, which are available in the **Tools/Outline** menu, you can perform very interesting operations on contours.

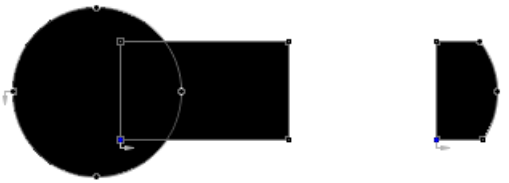
All three operations are applied to contours that have at least one node selected or to the whole glyph outline if nothing is selected.

The **Merge contours** command combines contours, removing all outline overlapping and keeping the filled result unchanged:



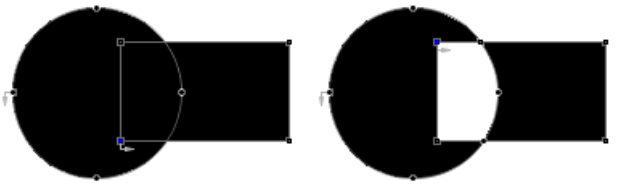
The **Merge contours** command is the outline equivalent of the Boolean "OR" operation.

The **Get intersection** command will keep only the area of intersection, removing all other parts:



This command is the outline equivalent of the Boolean "AND" operation.

The last command, the **Delete intersection** works in the opposite manner of the Get intersection command, keeping only those areas of contours that don't overlap:



This is the outline equivalent of the Boolean "XOR" operation.

Outline Optimization

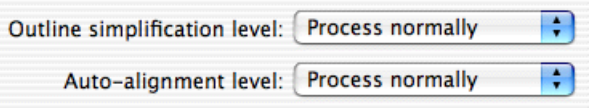
With the **Optimize** command FontLab tries to automatically adjust the outline to remove unnecessary elements and correct others.

To perform optimization use the **Optimize** command in the **Tools/Outline** menu.

Optimization does three things:

1. Removes unnecessary curve and line segments.
2. Aligns vertical and horizontal lines that are not precisely directed.
3. Corrects the connection types of lines and curves.

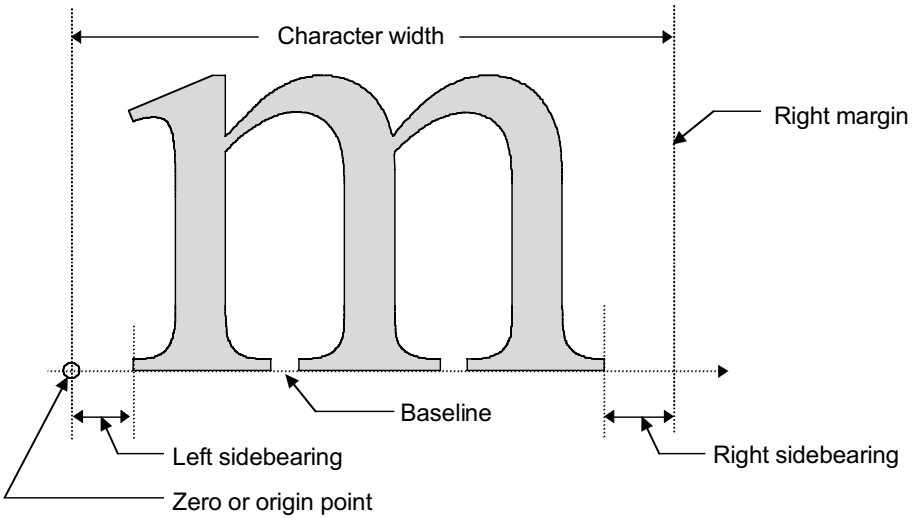
You can control the optimize features with the Font Audit section of the Preferences dialog box:



Outline Simplification level	Controls curve removal feature, from “do not simplify outline” to “extreme”. The bigger value you choose – the more curves FontLab will try to remove.
Auto-alignment level	Controls auto-alignment feature in a range from “do not align” to “extreme”.

Metrics

The Metric data of a glyph includes information about the horizontal and vertical width. Glyphs have an *origin point*, a *baseline*, *sidebearings*, and *left and right margins*:



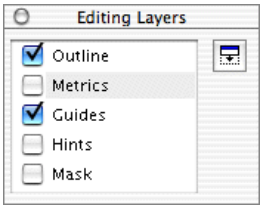
The baseline is used to align characters in a series. The left and right margins are used to define the positions of sequential characters in a series when the horizontal writing mode is selected. In the vertical writing mode the left and right margins are used to horizontally align characters and the top margin is used to vertically align characters.

In FontLab, the position of the origin point is the position of the left margin in the horizontal direction and the position of the baseline in the vertical direction. However, you can modify the position of any of the four margins. If you move the baseline or left margin line you will shift the entire glyph, but as a result the left margin is moved.

Editing Metrics

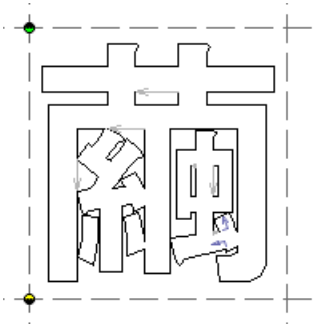
FontLab has a special window for editing glyph metrics, of course, but you can make small adjustments right in the Glyph Window, using the main Edit tool.

To change glyph metrics first activate the Metrics layer in the Editing layers panel:



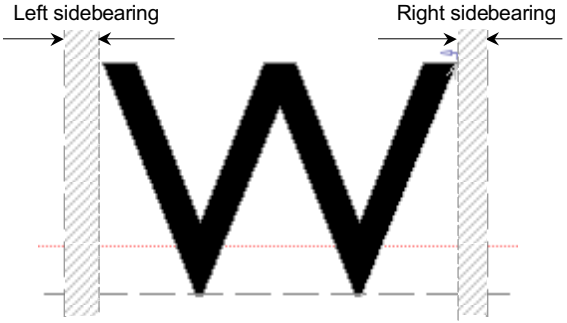
Then use the mouse and drag the left or right sidebearing or the baseline.

In FontLab you can define a vertical advanced vector for the glyph (this information is used in fonts which may type vertically, from top to bottom). To define a vertical advanced vector, press and hold the **SHIFT** key before moving the base line:

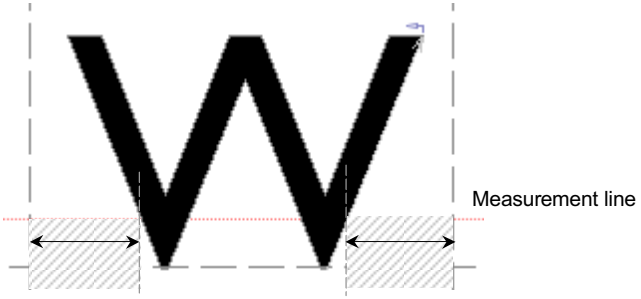


Using the Measurement Line

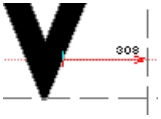
By default the sidebearing values are measured from the rightmost and leftmost points of the glyph:



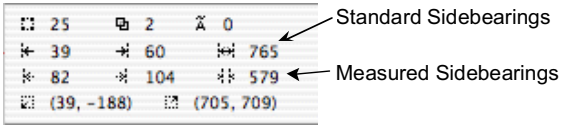
Sometimes you need to measure sidebearing values from some other point on the glyph outline. In FontLab you can do that by using the measurement line – a horizontal red line that defines the "height" of the sidebearing measurement:



You can see measured values appear above the line when you move the right or left sidebearing:



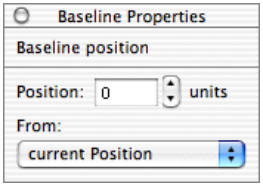
The same values appear in the glyph properties panel:



You can drag the measurement line to any position and the information in the Glyph properties will be updated accordingly.

Baseline Properties Panel

With this property panel you can adjust the position of the glyph's baseline. To open it **Ctrl**-click on the baseline and select the **Properties** command in the popup menu or just click the baseline while holding down the **Cmd** key on the keyboard.

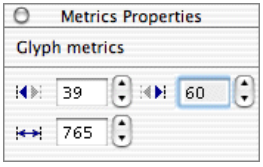


To change the position of the baseline:

1. Select the base level of the modification. It can be the old position (for relative offset) the top of the glyph, or the bottom of the glyph.
2. Change the position of the baseline relative to the base level.
3. Press the **Enter** key or click anywhere in the editing field to apply the changes.

Metrics Properties Panel

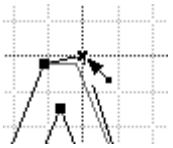
To open the metrics property panel, position the mouse cursor on the left or right glyph margin, **Ctrl**-click and select the **Properties** command, or just **Cmd**-click the mouse button on one of the margins.



In this panel you can modify a glyph's sidebearings and/or width.

Grid Layer

This layer is very simple: if the Grid is on, you will see a grid of vertical and horizontal lines in the Glyph Window. If snap-to-grid is on (which it is by default) any node that you move will snap to the gridlines.



You can adjust the grid frequency on the Glyph Window page of the Preferences dialog box (**Application/Preferences** command):

Guidelines Layer

Guidelines are straight lines that are used to guide the drawing of specific elements of a glyph. Guidelines can be vertical, horizontal or slanted.

Guidelines can be slanted at any angle from -45 to +45 from the vertical or horizontal direction. Slanted guidelines can help to mark *italic* characters, or specific slanted elements in normal characters, like the inner bar in the letter 'N'.

You can see little numbers giving the position and slanting angle of each guideline near the edges of the editing field of the Glyph Window where the guidelines cross the rulers.

There are *local* and *global* guidelines. Local guidelines appear only in the glyph where they were set. Global guidelines appear in all glyphs of the font. Global guidelines are very useful to mark important levels in the font (by using horizontal global guidelines) or to set the base direction of an italic or oblique font (using slanted vertical guidelines).

Editing Guidelines

Be sure that the Edit tool is active and the guidelines layer is visible – use the **View/Show layers** menu to switch it on. Note that the Guidelines layer will automatically switch on if you add a new guideline.

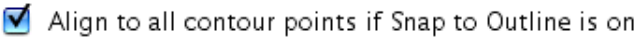
To add a new guideline:

1. Position the mouse cursor on the horizontal ruler bar (for a horizontal guideline) or on the vertical bar (for a vertical guideline).
2. Press the mouse button. The bar will appear "pressed" and the new guideline will appear.
3. Holding the mouse button down, drag the guideline to the desired place and release the button.

To move the guideline:

1. Move the mouse cursor onto the guideline that you want to move. Be sure that no other objects (such as nodes or hints) are near the cursor.
2. Press the mouse button and drag the guideline to the new place.

You may choose to activate the feature that will snap the guideline which you are moving or adding to all outline nodes. Use this option in the **Application/Preferences/Glyph Window** dialog box:



To slant the guideline:

1. Move the cursor onto the guideline near one of the sides of the editing field of the Glyph Window.
2. Press the mouse button. The mouse cursor will change to a pair of curved arrows that shows you the guideline slant direction.
3. Moving the mouse, slant the guideline to the angle that you want. Hold down the **Shift** key to constrain the slanting angle to 3-degree increments.

To remove the guideline:

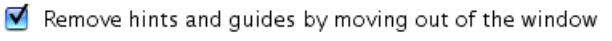
I.1. Start moving or slanting the guideline.

I.2. While holding down the mouse button, press the **Ctrl** key and then click the mouse button again.

II.1. Position the cursor on the guideline and **Ctrl**-click the mouse button.

II.2. In the menu, select the **Delete** command.

You can use this option:



located on the Glyph Window page of the Preferences dialog box, to remove any guideline or hint by simply dragging it from the editing field of the Glyph Window.

To remove all guidelines use the **Remove guidelines** command in the **Tools/Hints & Guides** menu. Options of this command include:

Both	to remove all guidelines
Vertical	to remove only vertical guidelines
Horizontal	to remove only horizontal guidelines.

The same commands are available in the rulers popup menu that appears if you **Ctrl**-click the vertical or horizontal ruler.

➲ **Note:** While you are editing the guideline, its parameters are shown on the status bar.

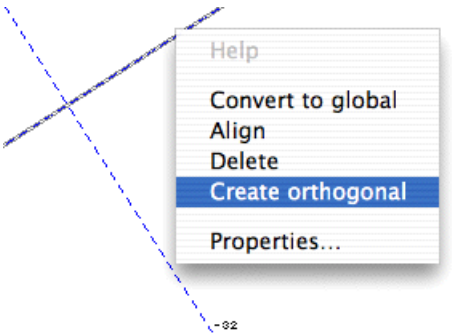
Guidelines Popup Menu

More commands are available in the guideline's popup menu.

There are two different guideline popup menus – one for local and one for global guidelines. Both menus include **Align**, **Delete** and **Properties** commands. The **Properties** command, as usual, will open the Property panel for the active guideline. The **Delete** command will remove the active guideline. The **Align** command is available only for slanted guidelines and will align them to the vertical or horizontal axis (i.e. remove their slant and make them vertical or horizontal guidelines).

Global guidelines have an additional set of commands located in the **Guideline Is** submenu. By selecting the commands in this submenu you can set the font parameters: **Ascender**, **Descender**, **Caps Height**, **x Height** or **Visual Ascender** and **Descender** that are used to set 100% zoom in the Glyph Window.

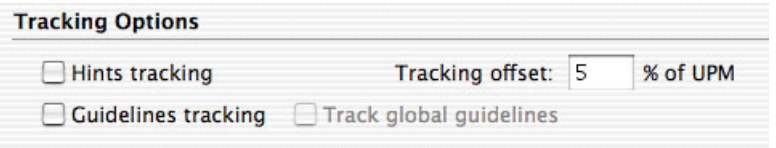
The local guideline menu has one additional command, **Create Orthogonal**, which you can use to create a guideline that is at a right angle to the current one:



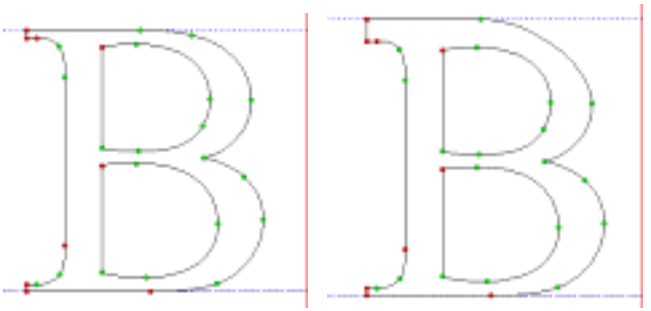
And naturally every local guideline can be changed into a global one by the **Convert to global** command.

Guidelines Tracking

There is a special feature that lets you modify an outline by moving guidelines. You switch this feature on in the **Glyph Window** page of the **Preferences dialog box** (choose the **Preferences** command in the **Application** menu to open the dialog box):



When guidelines tracking is active and you move a guideline a distance not greater than the **Tracking offset** parameter all outline nodes that were on the guideline will be moved with it:

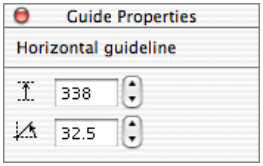


Original outline *Same outline after topmost guideline moved up*

If you move a global guideline and the **Track global guidelines** option is on, then all glyphs of the font that have nodes on this guideline will be modified. Be careful with this feature because the results of global guideline tracking are not undoable.

Guidelines Properties Panel

To open the guideline properties panel, **Cmd**-click the guideline or **Ctrl**-click it and choose the **Properties** command from the popup menu:



In this properties panel you can change the position and slant angle of a guideline. To apply the changes press the **Enter** key or just click anywhere outside the property panel.

Mask Layer

When you need something more than guidelines or a grid to help with glyph editing you can use the *mask layer*. The mask layer is an outline that is created with the same primitives as the glyph's outline. It appears in the Glyph Window as a dashed outline and the glyph's nodes "stick" to the mask. You can think of the mask as a "freeform" guideline.

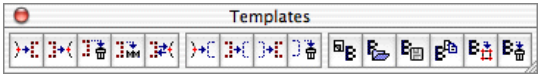
The mask layer is very useful when you want to use one font as a template for another font. For example, you can put the sans-serif version of the typeface into the mask layer while you are working on the serif version in the outline layer.

Another application of the mask layer is the creation of Multiple Master fonts. In this case you put one style of the typeface in the mask layer, another style on the outline layer and, after defining the design axis, you can put one of the masters on the mask, using the mask's "snap" feature or some other techniques that will be discussed later.

The mask layer can be filled in two ways: by copying the selected part of the outline to the mask layer or by using the **Assign Mask** command.

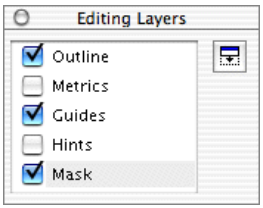
To copy the selected part of the outline to the Mask layer use the **Copy to Mask** command in the **Tools/Templates** menu. If nothing is selected in the outline layer the entire glyph outline will be copied.

To get access more quickly to the commands in the **Tools/Templates** menu, select the **View/Toolbars/Templates** option. It switches on the Templates toolbar:



Editing Mask

To edit the mask layer with the usual editing tools you need to activate it with the Editing Layers panel:



Alternatively you can use the **Edit Mask** command in the **View/Show Layers** menu.

When the Mask layer is selected for editing, the outline layer will be shown as a mask and may be filled, if Preview mode is active, so you can use it as a reference. Use any tool of the Edit mode to create, edit or remove the nodes and contours of the Mask layer outline. (Re)Activate the Outline layer when you are finished working on the mask.

Mask Operations

All operations related to the Mask layer appear in the **Tools/Templates** menu:

Paste Mask

Adds the contents of the mask layer to the outline. The added part will be selected so you can start to work with it immediately.

Clear Mask

Clears the mask layer, removing all its contents.

Mask to Master

Applies the Mask to Master action to the currently selected master. Refer to the “Multiple Master Fonts” chapter for more information about this action.

Exchange with Mask

Exchanges the outline layer and the Mask layer. Note the difference between this command and **Edit Mask** in the **View/Show layers**. When you use **Exchange with Mask** command, you actually exchange Mask and outline. The **Edit mask** command does this only temporarily and visually, for editing purposes only.

Assigning a Mask

With this command you can take glyphs from one font and put them into the mask layer of another font. The glyphs of the fonts are linked using their names, so the glyph with the name “zero” in the assigned font will be placed into the mask layer of the glyph with the name “zero” in the font where the mask is being made.

When you select the **Assign Mask** command from the **Tools** menu, you will see the dialog box:

There is a list of all open fonts in the top of the dialog box. Select the font whose glyphs you want to put into the mask layer of the current font (it may be the same font). If you select a Multiple Master font scroll bars and edit controls will appear allowing you to select an intermediate design of the font.

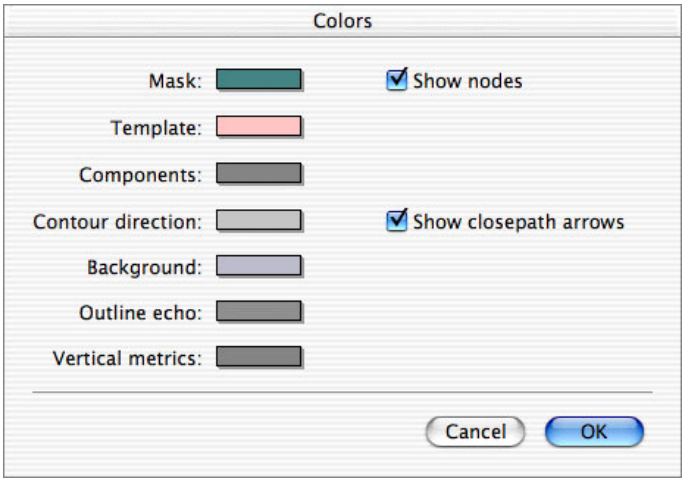
If you switch on **Create new glyphs if they do not exist in the font** option, then FontLab will create definitions for glyphs that are present in the “mask” font but absent in the current font. The newly defined glyphs will not have an outline but they will have a mask layer that you can use as a template.

Specifying Mask Appearance

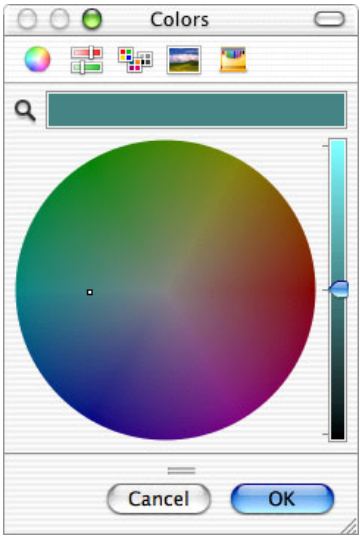
You can customize Mask layer appearance using the Glyph Window page of the Preferences dialog box (**Application/Preferences** command).

Open the Glyph Window page and press the **Colors...** button:

You will see a dialog box:



Using this dialog box you can adjust the color that is used to render various layers in the Glyph window. First color-selection field defines a color of the Mask layer. Click on the colored bar and the standard Color Picker window will appear:



Select the color you want to use to render mask on screen.

The check box to the right of the Mask layer color selector controls the appearance of the Mask layer's nodes. When this option is on, nodes on the mask layer can be seen:

The Mask layer without and with nodes

Template Layer

A template is a kind of mask that is font-wide and appears in all Glyph Windows. You could think of it as an additional **global** mask.

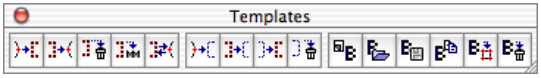
Template features and operations are very similar to those of a Mask: you can copy a selected part of any glyph to the template layer; copy a mask to the template layer; paste the template layer to the outline layer or clean the template layer.

If the snap-to-template function (View/Snap to layers) is on and the template layer is visible (View/Show layers), all nodes in all glyphs will snap to the template.

Commands related to the template layer are concentrated in the **Tools/Templates** menu:

Copy to Template	Copies the selected area of the glyph outline (or the whole outline if nothing is selected) to the template layer.
Copy Mask to Template	Copies the mask layer of the current glyph to the template layer. In other words, converts a local template into a global, font-wide template.
Paste Template to Mask	Copies the contents of the template layer to the mask layer of the current glyph.
Clear Template	Clears the contents of the font template layer.

To get access more quickly to these commands, select the **View/Toolbars/Templates** option. It switches on the Templates toolbar:



You can customize the color that is used to render the Template layer using the Colors dialog box accessible from the Application/Preferences/Glyph Window dialog box. Refer to the previous section for description of this dialog box.

Background Layer

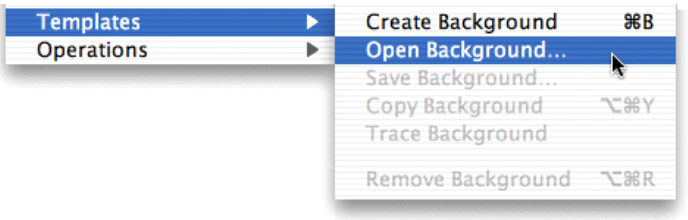
When other methods are not adequate you can use a *background bitmap* template. A bitmap template is a black-white bitmap image that appears on the screen underneath all the other layers. You can use it as a template for a glyph outline (it is especially useful when working with the Vector Paint tools) and you can automatically convert it into an outline if you have ScanFont 3.0 installed on the same computer.

Create a background layer using any of the three following methods:

1. Open a bitmap image file (in PICT or TIFF format).
2. Paste an image from the Clipboard.
3. Rasterize the current outline to make an image in the bitmap background layer.

You can also copy the contents of the background to the Clipboard to paste it into any Macintosh image-editing program; save it to the image file; and set its size and position on the screen.

To open a bitmap image, select the **Open Background** command from the **Tools/Templates** menu:



You will see the standard Mac OS Open File dialog box where you can select the bitmap file that you want to put into the background layer. Use the format selection popup menu to choose the format of the bitmap file. FontLab supports two formats: PICT (standard Macintosh bitmap format) and TIFF (standard image interchange format). Bitmap files that you import into FontLab must be black and white (line-art) images. Neither color nor grayscale images can be imported into FontLab. Your image editing application can usually make this change if need be.

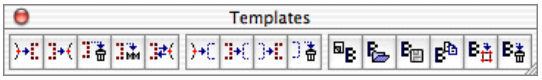
To copy a bitmap image from another Macintosh program into FontLab, select the image in the program using its selection tools; copy the image onto the Clipboard (the image may be color, black-white or grayscale); switch to the FontLab window; and select the **Paste** command from the **Edit** menu.

To rasterize a glyph's outline and make a background layer from it, select the **Create Background** command from the **Tools/Templates** menu.

Below is a table containing all the commands from the **Templates** menu related to background bitmap layer:

Create Background	Rasterizes the outline and makes a background layer.
Open Background	Opens a bitmap file (PICT or TIFF) from the disk and places the contents of the file into the background layer. You will see a standard Macintosh Open File dialog box when you select this command.
Save Background	Saves the background layer to a bitmap file (PICT or TIFF format). You will see a standard Macintosh Save File dialog box when you select this command.
Copy Background	Copies the contents of the background layer to the Mac OS Clipboard. You can also use the Paste command from the Edit menu to paste the bitmap contents of the Clipboard to the background layer.
Trace Background	Automatically traces the background layer and adds the tracing results to the outline. This action is available only if ScanFont is installed on the same computer. You can customize the autotracing options in the Trace page of the Preferences dialog box.
Remove Background	Removes the contents of the background layer.

To get access more quickly to these commands, select the **View/Toolbars/Templates** option. It switches on the Templates toolbar:

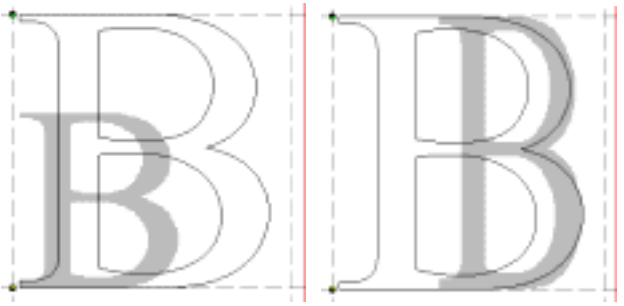


Note that if you apply the **Save Background** command to the glyphs selected in the Font Window (when the Font Window is active), then the backgrounds of all selected glyphs will be saved using filenames appended with the sequential number of the glyph, starting from 0.

You can change the color which is used to render the bitmap background in the Glyph window in the Colors dialog box accessible from the Glyph Window page of the Application/Preferences dialog box:

Background Positioning

This operation lets you set the size and position of the background layer:



Different sizes and positions of the bitmap background layer

To set size and position of the background layer:

1. Activate the Bitmap Positioning operation. Select the **Position Background** command from the **Tools/Operations** menu or press the button on the Tools toolbar or simply double click on the bitmap background while the Edit tool is active.
2. You will see a control box surrounding the bitmap.
3. Drag the handles in the corner of the control box to scale the background. Hold the **Shift** key to keep the proportions.
4. Position the mouse inside the control box, press the mouse button and drag the mouse to position the background.
5. Use the arrow keys on the keyboard or **Shift+arrow** keys to move the background.
6. **Ctrl**-click the mouse button to open a popup menu with the following commands:

Fit to glyph	Aligns the Bitmap background so it will fit the glyph outline.
Delete	Remove the contents of the background layer.

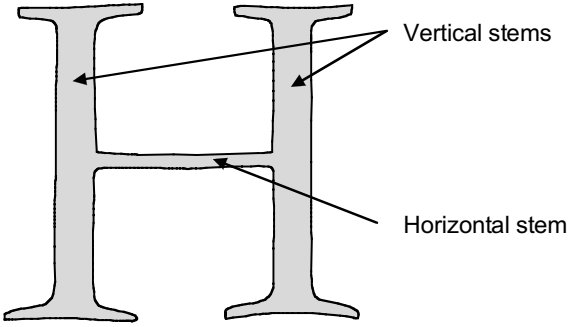
Double-click in the editing field or press **Enter** to finish changing the background or press the **Esc** key to reject it.

Hints And Links Layer

Hints are used by the font rasterizer to improve a glyph's appearance on devices with low output resolution, like computer monitors or low-res printers. A detailed description of the hinting methods and manual and automatic hinting features available in FontLab is in the "Hinting" chapter. For now we'll just mention that in the Glyph Window you can see font-level alignment zones and Type 1 character-level hints and links.

There are two hinting methods applied to Type 1 fonts (hints for True Type fonts are always generated automatically): font-level hinting and character-level hinting. Font-level hinting is generated automatically in FontLab, so you don't have to edit it manually.

Character-level hinting is applied to the characters' stems:



All important stems in a glyph should have stem hints, a pair of vertical or horizontal lines. The information in the hint includes not just the position of each of the two lines that "build" the hint, but also the position of one (major) line and the width of the hint.

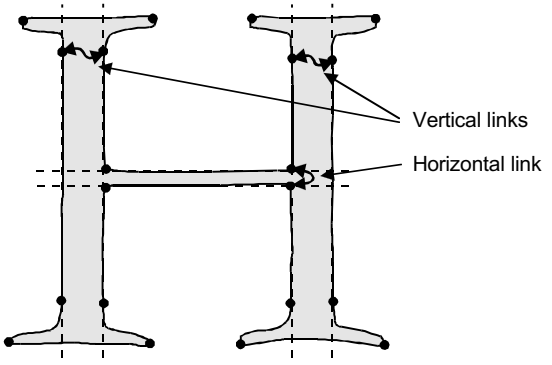
You can declare stem hints in FontLab just by dragging them and modifying their width. Because hints in FontLab are very "intelligent," they automatically snap to the contour, minimising your work. In most cases the autohinting algorithm that is included in FontLab produces good results – usually not any worse than the results of manual hinting.

There is a special feature, called **Hints Tracking** that can be used with hints. It will be described later, when we discuss editing hints.

Links

Stem hints are not connected to the outline – they exist in a different layer. This allows you to use hints as pairs of guidelines while you work on an outline. But if you change the outline after hints are set you have to set all the hints again to reflect the outline changes. Another kind of stem hint, called a *link,* may help in this case.

Links connect two outline nodes, using the position of the nodes and the distance between them, to define a stem hint. If you move one of the nodes connected by a link the link will automatically reflect the changes. Links can also be either vertical or horizontal:



Links are extremely useful when you are working with Multiple Master fonts. In these fonts each point has several “layers”, called masters, which represent different styles of the font. If you try to set hints in Multiple Master fonts, you will have to manually define the hints’ positions in each master. But with links you can just connect two outline nodes and hints will be generated automatically for each master when you export your font. Refer to the “Multiple Master Fonts” chapter for more information about links.

Editing Hints

Editing hints is very similar to editing guidelines. You can add new hints through the ruler bar of the Glyph Window; drag them with the mouse; and delete them by using the menu command or by **Ctrl**-clicking when dragging.

In contrast to guides, hints consist of two lines that can be moved together or separately. Hints cannot be slanted.

To add a new hint:

1. Position the mouse cursor on the horizontal ruler bar (for a horizontal hint) or on the vertical bar (for a vertical hint).
2. Press and hold the **Cmd** key. Press the mouse button. The bar will appear "pressed" and a new hint will appear. Release the **Cmd** key.
3. Holding the mouse button, drag the hint to the desired place and release the button.

To move a hint:

1. Move the mouse cursor onto one of the hint's lines.
2. Press the mouse button and drag the hint to its new place. Both hint lines will move together.

To move a hint's lines separately hold down the **Shift** key while dragging one of the hint's lines. Using this procedure you can change the width of the hint.

➲ **Note:** While you are editing the hint, its parameters are shown on the status bar.

To remove a hint:

I.1. Start editing the hint.

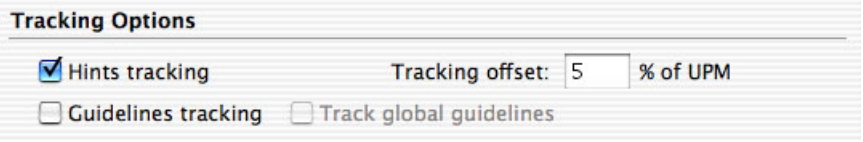
I.2. While holding the mouse button down, press the **Ctrl** key and then click the mouse button.

II.1. Position the cursor on the hint and **Ctrl**-click the mouse button.

II.2. Select the **Delete** command from the popup menu.

Hints Tracking

If the Hints tracking option of the **Glyph Window page** of the **Preferences dialog box** is on:



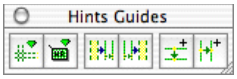
and you move hints a distance less than the **Tracking distance** setting, all nodes that are on the hint will be moved with it. Use this feature to keep an outline on the hint when you want, for example, to modify the hint's width. Note that this feature also works when you change a hint through the property panel described below.

Editing Links

Links connect two nodes. The only way to edit links is to connect them to different nodes of the outline.

To add a new link:

1. Select the **Add new Horizontal** (**or Vertical**) **link** command in the **Tools/Hints & Guides** menu or in the Hints & Guides toolbar:



2. Click the mouse cursor on the first node of the link.
3. Drag the mouse cursor to the second node and release the button.

To edit a link:

1. Position the mouse cursor on one of the link's lines.
2. Press the mouse button and drag the mouse. You will see that when you move the cursor onto the outline's nodes they become highlighted.
3. Position the mouse cursor on the node where you want to connect the link (it will become highlighted) and release the mouse button.
4. Release the mouse button while the link is not connected and a vertical link will disappear. A horizontal link will become a ghost link in this case (more on ghost links later).
5. **Ctrl**-click the mouse button while dragging a links' line to remove a link.

Hint and Link Popup Menu

To open the hint or link popup menu, CTRL-click one of the hint or link's lines.

The **Hint** popup menu includes the following commands:

Convert to Link	Converts the active hint to a link
Reverse	Reverses the direction of the hint
Delete	Removes the hint
Define Stem	Defines a vertical or horizontal stem (global hinting parameter) as equal to the width of the current hint
Properties	Opens the hint property panel.

The **Link** popup menu includes the following commands:

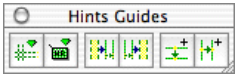
Convert to Hint	Converts the active link to a hint
Reverse	Reverses the direction of the link
Delete	Removes the link
Properties	Opens the link property panel.

Hint Commands

The **Tools/Hints & Guides** menu contains several commands related to hints:

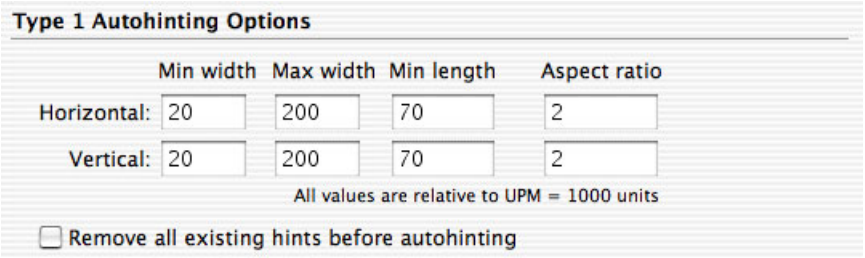
Remove Hints	Removes vertical or horizontal or all hints and links. This command is duplicated in the rulers context menu.
Autohinting	Automatically generates hints for the current glyph. Autohinting options can be adjusted in the Type 1 page of the Preferences dialog box.
Autoreplacing	Automatically generates a hint replacement program for the current glyph. Refer to the "Hinting" chapter for more information about hint replacement.
Convert Hints to Links	Converts all hints to links.
Convert Links to Hints	Converts all links to hints.

Same commands are duplicated in the Glyph Window context menu (which appears if you **Ctrl**-click in an empty area of the editing field). Some commands are available also in the Hints & Guides toolbar:



Autohinting Options

You can customize the autohinting algorithms using the Type 1 page of the Preferences dialog box (Application/Preferences command):

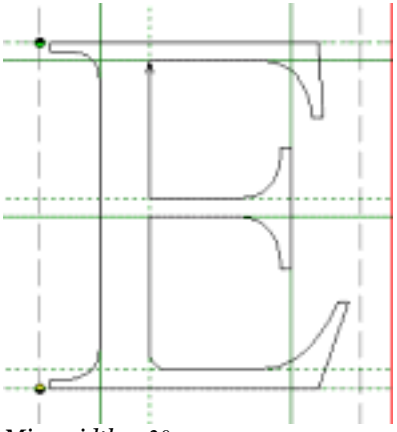


Width limits	Declares the minimum and maximum width of hints that the autohinting algorithm is allowed to create.
Min. length	Declares the minimum length of the nearest vertical (or nearest horizontal) vectors (or curve control vectors) that can be a candidate for building a hint.
Minimal length/ width aspect ratio	Declares a critical correlation between the width of a candidate for the hint and the length of the vector that builds that candidate.
Remove all existing hints before autohinting	Allows you to remove all existing hints before autohinting the glyph. If this option is switched off new hints will be added to the existing hint set. Of course, a hint substitution program will be built.

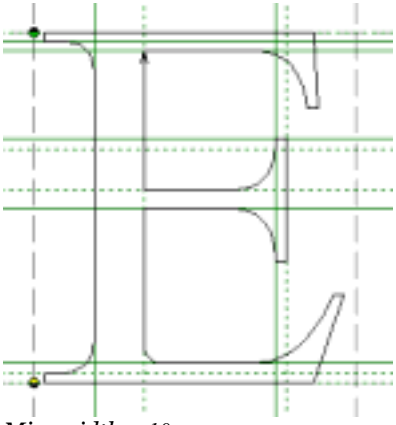
Note that all values are set for a 1000 UPM font and are automatically scaled by FontLab according to the real UPM value.

Some recommendations:

1. If you want FontLab to generate thin hints (horizontal serif hints, for example), set the **Min width** parameter to a value less then the width of the serif. Use the Meter tool to measure the width of the serif. Set **Min. width** to a value that is greater than the serif width to prevent FontLab from autohinting the serifs (the real width of the serif in the picture is 18 units):



Min. width = 20



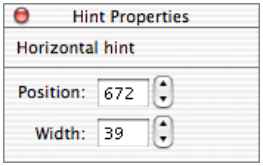
Min. width = 10

2. Decrease the **Min. length** value to generate more hints. Increase the value to generate only the most important hints.

3. Increase the **Max. width** value if you are working on a black or heavy font that has very thick stems.

Hint Properties Panel

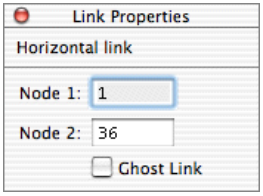
To open the hint property panel, **Cmd**-click one of the hint lines or **Ctrl**-click and choose the **Properties** command in the popup menu:



In the hint properties panel, you can modify the position of a hint in the upper edit box and modify the width of the hint in the lower box. Press the **Enter** key or click the mouse outside of the properties panel to apply the changes.

Link Properties Panel

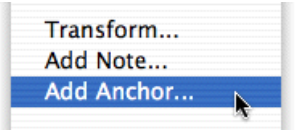
In the Link properties panel you can modify a link's properties: enter the index numbers of the two nodes which you want to "connect" by the link in the editing fields or check the **Ghost Link** checkbox to make a link into a ghost link:



Anchors Layer

Sometimes when you work with a glyph it's helpful to mark particular positions and refer to them later. In FontLab you can use a special editing layer called Anchors to do this. Anchors are simply named points that you can add, remove, move or rename. A special property of anchors is that you can use them in macro programs and in the automatic glyph construction feature.

To add an anchor, **Ctrl**-click anywhere in the empty space of the editing field and select the **Add Anchor** command in the popup menu:

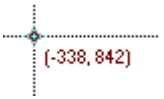


You will see a new anchor (by default it is a small red cross) and an editing field asking you to enter a name for the anchor:

Name the anchor and click the **Enter** key to complete.

Moving Anchors

To move an anchor, just drag it with the Edit tool. You will see the anchor position highlighted while you are dragging it:



Removing Anchors

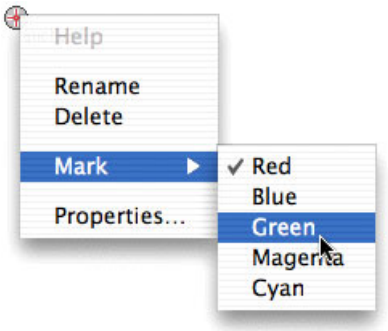
To remove an anchor, **Ctrl**-click it and select the **Delete** command in the popup menu.

Renaming Anchors

To rename an anchor, **Ctrl**-click it and choose the **Rename** command. Change the anchor name in the editing field.

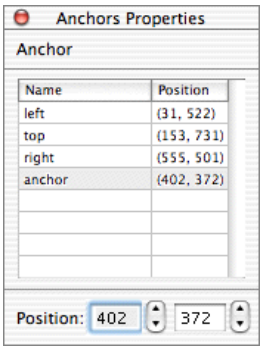
Changing Anchor Color

You may choose one of five colors to mark the anchor. **Ctrl**-click the anchor; select the **Mark** command from the popup menu; and choose the color:



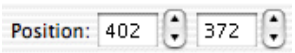
Anchor Properties

Cmd-click the anchor or **Ctrl**-click and choose the **Properties** command in the context menu to open the anchor Properties panel:



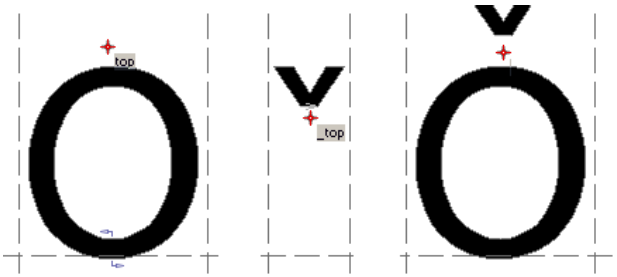
This panel lists all anchors defined for the glyph. To change the properties of one of the anchors, select it in the list and the anchor will be highlighted.

To change the position of the anchor, use the editing fields below the list:



Using Anchors to Build Composites

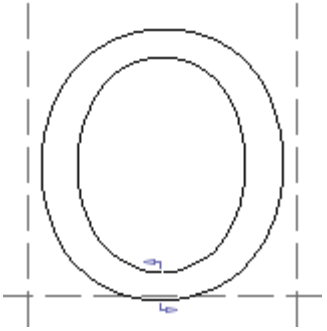
The main purpose of anchors is to mark important positions in the glyph space. In addition, anchors may be used to “link” some glyphs and form composite glyphs:



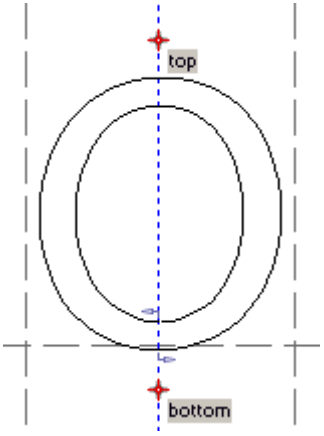
You can define a composite glyph using the Composites tool any time, but that operation creates “fixed” composites and you would need to perform it manually for every composite glyph you want to create. For composites that consist of a base glyph and one or more “accent” glyphs this operation may take a lot of time.

In contrast, with anchors you can create “virtual” composites, which can be converted to the fixed state at any time with the **Glyph/Generate Glyphs** command.

To define a virtual composite the first thing you need to do is to open the base glyph:



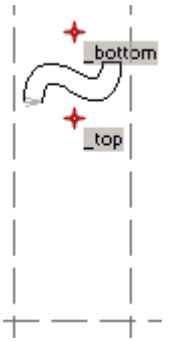
Then you place one or two “link” anchors, which must be named “top” and “bottom”, at the points where accents will be attached:



To add anchor points use the **Add Anchor** command from the Glyph Window context menu or the keyboard shortcut: **Ctrl+Option+Shift**+click. The shortcut will first add the anchor named “bottom” and then “top”.

The next step is to add matching anchors in the accent glyph. Open one of the accent glyphs:

Add an anchor named “_bottom” at the point in the accent glyph where it will connect with the “bottom” anchor in the base glyph. Add the anchor “_top” where it connects to the “top” base anchor:

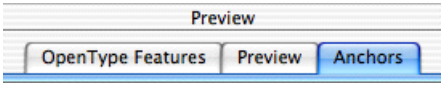


You may use the **Add anchor** command to add these anchors or the keyboard shortcut: **Ctrl**+**Option**+**Shift**+click will first add the “_bottom” anchor and then the “_top” anchor.

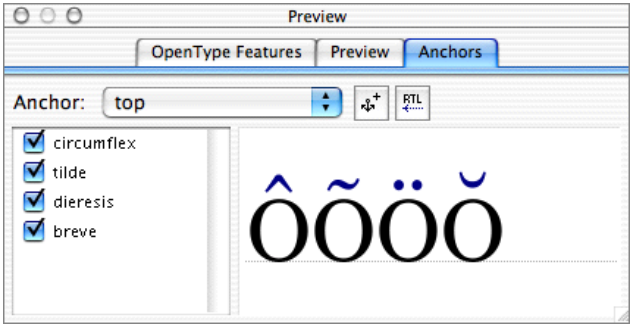
Using the Anchors Panel

Use the Anchor panel to preview “virtual” composites:

1. Open the Preview panel (**Window** menu or the button in the Panels toolbar).
2. Select the Anchors page of the preview panel:



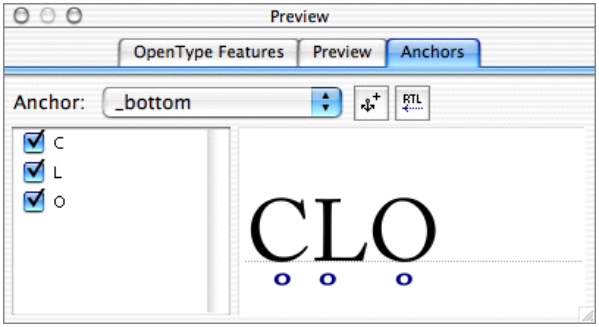
3. Open the base glyph in the Glyph Window. The Anchor panel will immediately list all of the anchors and all matching accent glyphs that have “link” anchors:



The left list includes all accent glyphs that can be linked with the base glyph using one of the defined anchors. The first 10 resulting virtual composites are previewed in the sample string. Use the check boxes to the left of the accent list to choose which composites to preview:

4. Choose another anchor in the list to update the accent list with accents that can be connected to that anchor:

5. Double-click one of the virtual composites to select one of the accented glyphs for preview:



As you can see, now the anchor list includes all virtual composites that can be created using the selected accent glyph and the selected anchor. Double-click one of the virtual composites to select it for previewing.

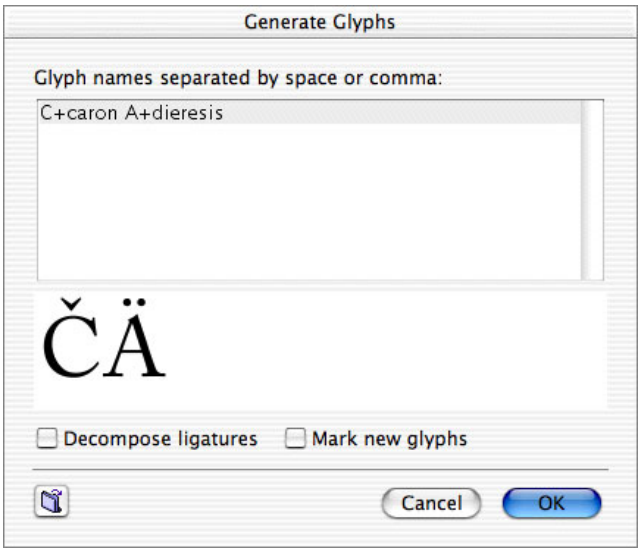
6. If you move anchor point currently selected in the Anchors panel with the Edit tool you will see the sample string automatically updated to show the new shapes of the virtual composites.

The panel contains two buttons to the right of the anchors list:

	Click this button to add a new anchor to the glyph that is open in the Glyph Window.
	When this button is pressed, the anchors panel previews text in right-to-left mode, which is needed if you are working with Arabic or Hebrew glyphs.

Creating Composites and Ligatures

To create real composite glyphs from the base and accent glyphs use the **Generate glyphs…** command from the **Glyph** menu:



In the dialog box enter the composite name using the simple syntax:

[base glyph name]+[mark glyph name]

for example:

C+caron A+dieresis

You may enter more than one name separated by a space or comma.

With the same dialog box you may create ligature glyphs, like “fl, ffl” or others. Just enter the name of the ligature using the simple syntax:

[first component name]_[second component name]_[third component name]_[…]

For example:

f_l, f_i, c_t

Check the **Decompose ligatures** option below the preview field to “paste” all the components’ outlines and make a decomposed glyph:

If you click on the button you can choose and open the text file with glyph name definitions prepared in advance.

Check the **Mark new glyphs** option to mark created glyphs in the Font window with color.

Aliases Table

FontLab comes with an alias.dat file (located in the FontLab/Mapping folder), which is the text file that defines the decomposition of common composite glyphs. Here is a part of the alias.dat file:

```
%%FONTLAB ALIASES
nbspace space
hyphen minus
periodcentered middot
onesuperior one
ordmasculine o
onequarter one_slash_four
onehalf one_slash_two
```

As you can see, every line of the file contains the real name of the component glyph followed (with a single space as the separator) by the decomposition name:

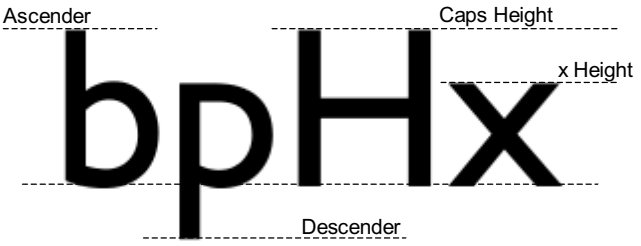
```
onequarter one_slash_four
```

This means that when you request the glyph named “onequarter” in the Generate Glyphs dialog box, FontLab will create a new glyph named “onequarter” but built as a ligature using the “one”, “slash” and “four” components as if you had entered the name “one_slash_four”.

You can modify this table to include glyph names you often need to create.

Vertical Metrics

Every font has several vertical metrics values for alignment of text:



The **Ascender** line defines the position of the top of lowercase characters (usually the topmost point of the Latin ‘b’).

The **Descender** line defines the position of the bottom of the lowercase characters (usually the bottom point of ‘p’).

The **Caps height** defines the height of the uppercase characters (without overshoot), usually it is the height of ‘H’.

The **x Height** is the height of most lowercase characters, like ‘x’ or ‘v’.

In FontLab you can modify the vertical metrics values in the Font Info dialog box, but you can also preview and change them visually in the Glyph Window.

Make sure that the Vertical metrics layer is active and not locked.

In the Editing field vertical metrics appear as gray lines with a label at the left:

- Ascender
- Descender
- Caps height
- x Height

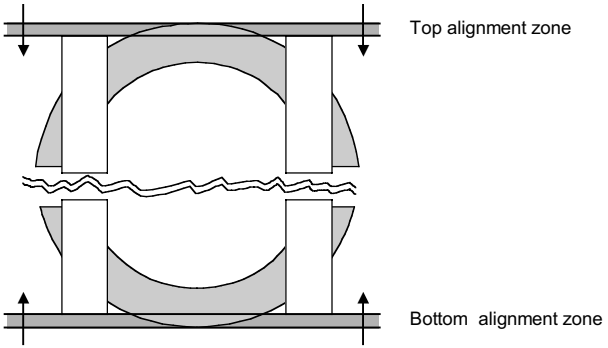
To change a metric, just drag its line with the Edit tool.

To change the color of vertical metrics lines, refer to the Colors dialog box - accessible with the **Colors...** button on the Glyph Window page of the Preferences dialog box.

Alignment Zones

Most fonts have “square” and “round” glyphs. Round glyphs (like “O” and “Q”) usually have 3-4% “overshoot” on the bottom and top. “Overshoot” is the amount that a glyph extends above or below its nominal top or bottom. It is used to optically correct the appearance of “round” glyphs, which tend to appear too small at their nominal height.

When a font is rendered on a device with limited resolution, it is often necessary to “suppress” the overshoots to make a line of text look smooth:



This whole process is described in full detail in the Hinting chapter. Here we will only discuss the basic modification of the layer with the Edit tool.

To let a font-rendering program perform overshoot suppression you need to “declare” overshoots using alignment zones:

You can modify alignment zones with the Edit tool: just drag the bottom or top line of the zone to change its width or position.

You may later use the FontInfo dialog (File/Font Info) to check the exact parameters of the zones.

Sketch Mode

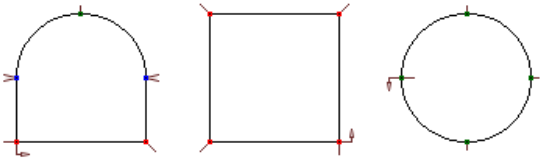
Sketch mode is a new easy way to create outlines.

When you enter sketch mode for the first time the glyph outline is automatically converted into a special format that is optimised for “freestyle” editing. Thus the sketch mode has two outline layers: one with the original “real” glyph outline, and one with the sketch outline.

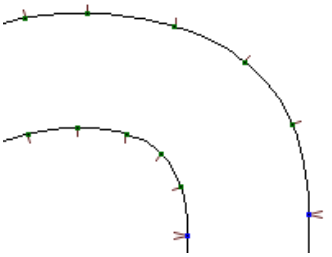
In contrast to the outline representation used in the Edit mode, the Sketch outline is not WYSIWYG. Before exporting to a font file all sketch outlines must be converted into “standard” outlines. I.e. you edit the sketch, but nothing changes in the real glyph outline until you do the conversion.

The Sketch outline is entirely point-based – all points that define the outline are positioned on the outline. There are no Bezier control points or TrueType off-curve points:

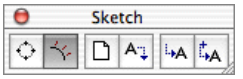
There are 3 types of points: corner, curve and tangent:



When points of known types are set, FontLab automatically “draws” the outline to match their positions. If you want to change the shape of the outline, just move the points or add more points to increase the precision:



When you enter the Sketch mode, you will see an additional Sketch toolbar:



The buttons on the Edit toolbar are available too and have the same meaning as in Edit mode. Here is a short description of the commands available on the Sketch toolbar:

	Show outline	Displays the real glyph outline in the background of the sketch outline.
	Show marks	Shows point and direction marks.
	New sketch	Used to create a new, empty sketch. I.e. delete the current sketch.
	Import sketch	Converts the glyph outline to a sketch.
	Replace outline	Replaces the glyph outline with the sketch.
	Add to outline	Adds the sketch outline to the glyph outline.

The same commands are available in the popup menu that appears if you **Ctrl**-click the empty space of the Glyph window.

Visualization of the Sketch Outline

Every point in Sketch mode can have 2 marks:

The Node icon, which is similar to how nodes are shown in Edit mode.

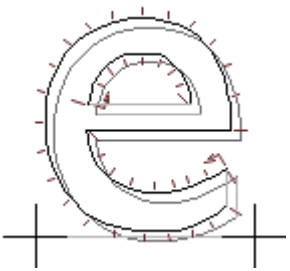
The Direction mark which shows the curve direction (the mark is on the convex side of the curve) at the point.

Direction marks are different depending on the type of the node:

Short mark		Curve point
Longer mark		Corner point
Double mark		Tangent point
Mark with an arrow		Starting point of the contour

Use the **View/Show Layers/Nodes** ( ) command to switch the node icons on or off and the **Show Marks** ( ) button to show or hide the direction marks.

With the Show Outline button you can turn the real (Edit mode) glyph outline in the background of the Sketch outline on or off:



The glyph outline may appear contoured or filled, depending on the preview mode.

Moving Points

To move points simply drag them with the mouse button. Hold the **Shift** key down to constrain movement to the vertical or horizontal direction.

Changing Point Type

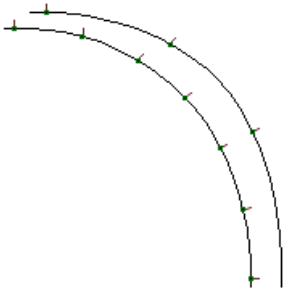
To change point type from corner to curve to tangent simply double-click the node with the mouse button.

Removing Points

There are two ways to remove unnecessary points:

1. Use the Eraser tool. Select the tool (), position the cursor over the point you want to remove and press the mouse button. Drag the mouse to remove multiple points. **Cmd**-click and drag to surround the points you want to remove with a rectangle.
2. Using the Edit tool (), start dragging the point, press the **Ctrl** key and then click the mouse button without releasing the **Ctrl**.

When you remove a point, you'll notice that FontLab tries to keep the outline smooth:



In normal outline mode you usually try to minimise the number of nodes and use Bezier control points and off-curve points to make the outline look smooth. The main feature and benefit of Sketch mode is that you can add as many curve points as you want to make the outline smooth and precise, and the curves will be aligned and optimised automatically.

Inserting Points

One way to insert points is to use the Knife tool ( ).

1. Select the tool on the Edit toolbar.
2. Click on the outline where you want to insert a point.

Alternatively you can add points with the Edit tool:

1. Position the cursor on the outline.
2. Press the **Option** key and click the outline.

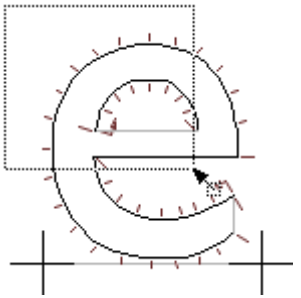
Reversing Contours

Cmd-click one of the contour's points to reverse its direction.

Selecting Points

As in the standard Edit mode, you can select many points and perform group operations, like moving the selection or copying to and from the Clipboard.

To select points with the Edit tool, press the mouse button in an empty area and drag the cursor to surround the points you want to select with the rectangle:



Hold the **Shift** key down to invert the selection state of the points.

To select the whole contour, double-click it. Note that you must not double-click one of the points – this operation will change its type.

Alternatively, hold down the **Cmd** key and click anywhere near the contour.

You can use commands from the **Edit** menu to select or deselect parts of the outline:

Select All	to select the whole outline
Deselect	to remove all selections
Invert Selection	to invert the selection state of all nodes.

Click anywhere in the empty area anytime to remove all selections.

Using the Magic Wand Tool

With this tool you can easily select multiple contours. It is especially useful when your sketch contains multiple overlapping outlines.

1. Activate the Magic Wand tool ( ).
2. Click anywhere near the contour you want to select. The closest contour to the point where you clicked will be selected.
3. Hold down the **Shift** key to reverse the selection state of the contour.

Moving the Selection

When multiple points are selected you can move the entire selection with the Edit tool: just click on any selected point or selected part of the outline and drag the mouse. Hold down the **Shift** key to constrain movement to the vertical or horizontal direction only.

Transforming the Selection

You may rotate, scale, and slant a selection with the Rotate ( ), Scale ( ) and Slant ( ) tools on the Edit toolbar.

1. Select the part of outline you want to transform or remove all selections to transform the whole outline.
2. Choose the tool to perform the transformation.
3. Click the mouse button on the center of transformation and drag the mouse to rotate, scale or slant. Hold down the **Shift** key to constrain the transformation.

Selection Operations

You may copy, paste, delete and duplicate a selected part of the Sketch outline. Use the following commands from the **Edit** menu:

Copy	Copy the selection to the Clipboard
Paste	Add the Clipboard contents to the current sketch
Delete	Remove all the selected points
Duplicate	Add a copy of the selection.

When you use the **Paste** or **Duplicate** command, the selection is added with a shift from the original position. You may control the amount of this shift on the Glyph Window page of the Preferences dialog box (**Application/Preferences**).

Breaking and Joining the Sketch Outline

You may break the sketch outline at any point with the Knife or edit tool.

To break the outline with the Knife tool, select the tool ( ) and click on the node where you want to make a break.

To break the outline with the Edit tool, hold the **Option** key and click the point where you want to make a break.

To join two contours, position the starting or ending point of the first contour over the ending or starting point of the second contour.

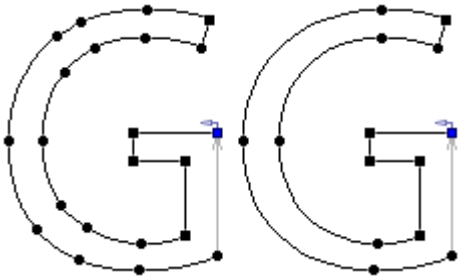
Hold the **Option** key down while moving the point **to prevent the contours from joining**.

Converting to Outline

Click the **Replace outline** button to convert the Sketch into a normal outline. FontLab will try to optimise the result using the minimum number of curves while maintaining high precision.

With the **Add to outline** command you can add the contents of the Sketch outline to the normal glyph outline.

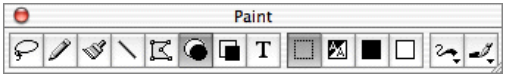
To see the converted result simply change to Edit mode ( ). If you think that the Sketch converter produced too many curves, use the **Optimize** command in the **Tools/Outline** menu to fix it:



VectorPaint Mode

Vector Paint is FontLab's unique set of tools that allow you to paint vector contours with tools that look and feel like bitmap tools. You can choose brushes, pens, freeform selections and even enter text. The idea of Vector Paint is that all the tools produce contours that combine with the existing glyph contours using our unique contour-processing technology.

When you click the Vector Paint button in the toolbar, a Vector Paint toolbar will appear where you can select one of the Vector Paint "sub-tools":



If the **Do not open Glyph toolbox and VectorPaint toolbar** option is selected on the Glyph Window page of the Preferences dialog, then the Vector Paint toolbar will not appear. You should use the Edit toolbar (which contains a subset of painting tools) in this case.

The type of interaction between existing and new contours depends on the selected color mode. This process is very fast and is completely transparent to you, so if you switch on the preview mode (where the glyph appears filled), the illusion of bitmap-like editing of a contour-based glyph image is very realistic.

All the tools can work in 4 different color modes:

Transparent	Newly created vector objects that are generated by the application of Vector Paint tools do not interact with the existing glyph's contour and appear selected for easy editing.
Automatic	The color of the brush depends on the point where you begin drawing. If you begin in a white area, a white brush will be selected, if in black, a black brush will be selected. Use this color mode to easily extend white or black areas of the glyph.
Black	Generated contours are added to existing contours, expanding the black area of the glyph. It looks like a black brush applied to a black picture.
White	New contours are subtracted from existing contours, simulating a white brush or eraser.

Here is an example of a brush stroke applied with Transparent, Black and White “colors”:

➲ **Note**: Vector Paint tools have an option to automatically activate the Transform Operation when any of the painting operations is completed. This option allows you to instantly move, scale, rotate or slant the newly created contour.

Here is a list of all available Vector Paint tools with a short description of each:

Select	Used to select non-rectangular areas of a glyph. It selects not the nodes, like the Edit tool, but actually cuts lines and curves and selects black areas that can be moved or otherwise transformed.
Pen (Contour)	Used to create new contours or modify existing ones. It is not really a “paint” tool, because it deals with contours, but it is a very natural and flexible tool used to adjust the result of the application of Vector Paint tools.
Brush	Exactly that – a brush. It can be round or calligraphic. A calligraphic brush can be of any size and slant angle.
Line	Used to draw straight lines with a selected brush.
Polygon	Has two modes: point-by-point polygon drawing with easy combination of vectors and curves, or point-by-point definition of a polygon that will be drawn by using the selected brush.
Ellipse	Used to draw ellipses or circles.
Rectangle	Used to draw rectangles or squares.
Text	Used to enter text (vector based) using any TrueType font installed in the system.

Freehand Select Tool

This tool works like a precision knife. You can cut part of a contour, and it will be automatically selected so you can transform it, delete it or copy it.

To select part of a glyph with the freehand select tool:

1. Select the freehand select tool ( ) in the Vector Paint toolbar.
2. Position the cursor on the point where you want to start the selection and press the mouse button.
3. Drag the mouse to extend the selection polygon in freehand mode, or click the mouse button to extend the selection polygon by straight vectors.
4. **Ctrl**-click to finish the selection.

When you finish the selection, you will see that the selection polygon was applied like a knife and you have a new contour (or several contours) that is separated from the glyph. The new contours are selected so you can use the Edit tool to move them or begin a Transform operation and transform the selection. Of course, you can use any **Edit** menu command with this selection.

Pen (Contour) Tool

With the Contour tool you can create new contours or modify existing contours in a more artistic manner than with the Edit tool. When you use the Contour tool, you can draw new contours just as you do on paper. FontLab will trace your drawing and replace it with a series of curves and lines.

How to create a new contour

If you begin a contour in a free area (where the cursor has its ordinary shape), you will define a new contour. If you want to begin a new contour but its starting point is on an existing contour press the **Cmd** key to force FontLab to create a new contour.

How to modify an existing contour

When you move the cursor of the Contour tool onto an existing contour or node, it changes. If you begin drawing (without holding down the **Cmd** key) the new contour will be inserted into the existing one. If the finishing point of your drawing is on an existing contour also, and the starting and finishing points are on the same contour, then the new drawing will replace the part of the existing contour that lies between the starting and finishing points.

How to draw a single curve

Hold the **Shift** key down when you release the mouse button after drawing a new line. Your drawing will be approximated by a single curve. This is a good way to draw a new contour step-by-step.

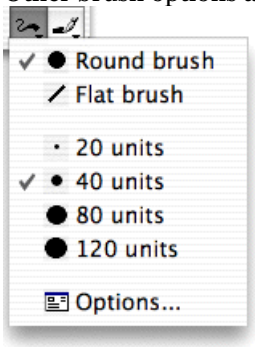
Brush Tool

The brush tool works like the usual bitmap brush that you find in any bitmap-editing program. You begin a brush stroke by pressing the mouse button. Draw the stroke by dragging the mouse and finish drawing it by releasing the mouse button.

To change the color of the brush, use the color selection buttons on the Vector Paint toolbar:

	for the “empty” color
	for the “auto” color
	for the “black” color
	for the “white” color

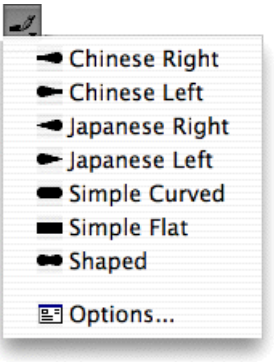
Other brush options are accessible in the Vector Paint options menu:



You can paint with round or calligraphic brushes of different widths:

	Round brush
	Calligraphic-style brush
	20 unit wide line
	40 unit wide line
	80 unit wide line
	120 unit wide line

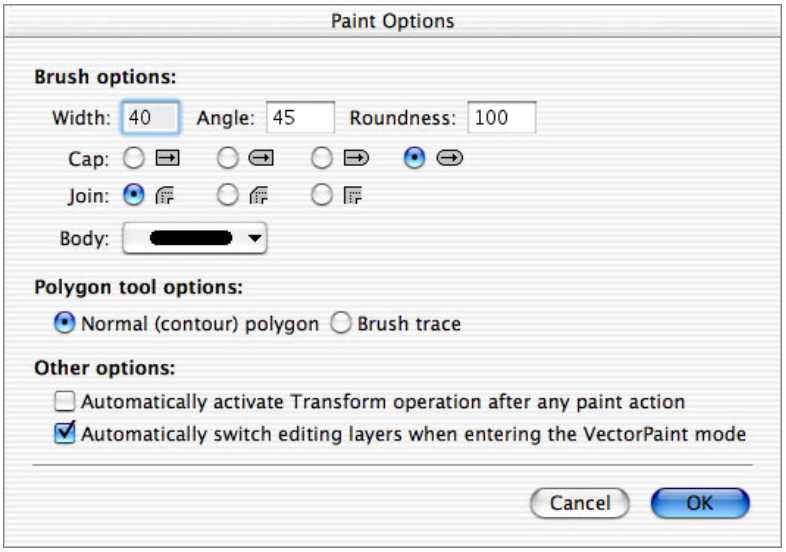
You can also specify a brush stroke shape. Press the button and select a shape in the popup menu:



This is an example of different brush strokes:

VectorPaint Options

You can also change the brush properties in the Paint Options dialog box. **To open the Paint Options dialog box**, choose the **Options** command in the brush options menu (). You will see following dialog box:



In the **Paint Options** dialog box you can enter the width of the brush and change the slant angle of a calligraphic brush. Additionally, you can select how the brush strokes are started and finished. Choose the brush's starting and finishing shape by activating one of the radio buttons.

You can also select the style of the connection between two sequential segments of brush strokes. It can be sharp, smooth or flat. Select one in the **Paint Options** dialog box. Icons near the radio buttons give an explanation of the styles of connections.

The **Body** popup menu in the Brush options area lets you to select the shape of the brush stroke.

Other options relate to the Polygon tool (described later) and the option for automatic activation of the Transform operation.

To choose one of the Polygon tool modes use the options in the Polygon tool field. If you choose **Normal (contour) Polygon** the polygon tool will create a simple contour that can include vectors and curves. If you choose the **Brush trace** option the current brush will be applied to the created polygon's contour. In the Brush trace mode you cannot draw curves while defining a new polygon.

If you mark the **Automatically Activate Transform Operation** check box and select the Transparent painting color, the Transform operation will be activated and applied to the contour that you created after the completion of any paint action.

The last option, **Automatically switch editing layers...,** allows you to automatically switch editing layers when you enter the Vector Paint mode. To set this option, first turn it on; then enter Vector Paint mode for the first time and switch editing layers to create the most comfortable environment. For example, in Edit mode you may use an unfilled outline with all nodes switched on. In Vector Paint mode you may switch on the preview mode and switch off all nodes. Now when you enter Vector Paint mode the next time, the editing layers will be switched automatically to your preferred environment.

Line Tool

The Line tool allows you to apply brushes to straight-line segments.

To draw a line segment:

1. Position the mouse cursor on the beginning point and press the mouse button.
2. Move the mouse to the end point and release the button. Hold down the **Shift** key to constrain the direction of the line to 15-degree increments.

Polygon Tool

The Polygon tool can be used in two modes: as a tool to draw a polygon consisting of lines and curves or as a tool to draw an outline of a polygon with a selected brush. The second mode can be treated as a series of applications of the line tool. The mode of the polygon tool can be selected in the **Paint Options** dialog box.

To draw a polygon using the Polygon tool:

1. Select the polygon tool in the Vector Paint toolbar. Be sure that the polygon tool is in the polygon mode.
2. Move the mouse cursor somewhere in the editing field and press the mouse button. Holding down the button, move the mouse cursor to the first point of the polygon. Release the button.
3. Move the mouse cursor to the position of the next polygon point. To add a line segment, click the mouse button. To add a curve segment, press the mouse button and drag the mouse to define the control vector of a curve. Hold down the **Shift** key to constrain the direction of vectors to 15-degree increments.
4. If the polygon in its present state is finished by a curve, the next segment that the polygon tool will try to add will be a curve. To switch between adding a vector or a curve use the **Tab** key on the keyboard.

   Using the '8', '9' and '0' keys on the keyboard you may choose the type of the next segment to add. Click '8' to append a line, '9' to append a smooth curve and '0' to append a sharp curve.
5. Repeat steps 3 and 4 for all points of the polygon.
6. **Ctrl**-click the mouse button to finish creating the polygon.

The **Brush Trace** mode of the polygon tool works just like the normal (contour) mode.

Ellipse and Rectangle Tools

The Ellipse and Rectangle tools are very similar. The only difference is in the result.

To draw an ellipse or rectangle:

1. Select the tool that you want to use (or).
2. Position the mouse cursor on the spot where you want to place one of the rectangle corners (or on one of the corners of the rectangle that surrounds the ellipse).
3. Press the mouse button and drag the mouse to define the rectangle (or ellipse).
4. Hold down the **Shift** key to constrain the direction of the rectangle's diagonal to 15-degree increments.
5. Release the button to finish creating the rectangle (ellipse).

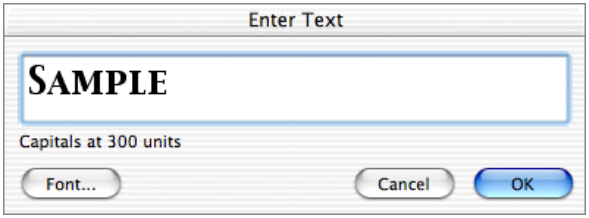
Text Tool

With the Text tool you can add text to a glyph. Carefully select the color mode when planning to use the text tool. It is usually best to use the "Empty" color because in that mode the text stroke will not interact with the existing contour and you will be able to adjust its position using the Edit tool or Transform operation.

Select the **Automatic Activation of Transform Operation** option in the Paint Options dialog box. With this option on the Transform operation will be activated immediately after entering the text string, allowing you to modify its size or position.

To enter a string of text:

1. Select the text tool T in the Vector Paint toolbar.
2. Position the mouse cursor (with the crosshair and the "suggested rectangle" of the future string) on the place in the editing field where you want to add the string.
3. Click the mouse button.
4. In the dialog box, enter the character string. Use the **Font** button to select the font that will be used:



   Below the sample string you will see the name of the current font and the size of the text string. The size is presented in font units. You can change the string size in the Font dialog box. The size of the placed text will be 10 times the selected point size. For instance, if you select a 24 pt. font you will get a string that will be 240 units in height.
5. Press **OK** to enter the string or **Cancel** to abort this operation.

Meter Mode

With this tool you can measure any distance and angle in your glyph. It is very useful if you want to create very precise, extremely high quality symbols.

To measure distances between two points:

1. Select the **Meter** tool in the Tools toolbar. The **Meter** bar appears in the top part of the Glyph Window:

This is a brief description of the fields on the Meter bar:

-22,602	Absolute position of the point (relative to the glyph zero point)
-22,602	Reference distance (relative to the position of the reference point)
326.2	Horizontal and vertical distance (from the beginning to the end of the metering line)
34.8	Geometric distance
268,186	Angle of the metering line.

Note that you can switch the Meter bar on or off at any time if you click on the button in the bottom-left corner of the Glyph window.

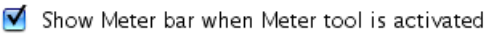
2. Position the mouse cursor on your first point.
3. Press the mouse button and drag the mouse to the second point. In the Meter bar you will see the vertical, horizontal and direct distance between two points and the angle of the vector connecting these points. Hold down the **Shift** key while you drag the mouse to constrain the measurement to 15-degree increments.

While you are dragging the mouse you will see that the **Meter** tool sticks to any object that it can find in the editing field.

To measure the distance from a contour:

1. Put the mouse cursor on the contour from which you want to measure.
2. Press the mouse button and drag the mouse to what you want to measure to. Hold down the **SHIFT** key and the direction of the mouse's movement will be constrained to the normal direction of the contour starting point.
3. When you're done, release the button.

Note that the Meter bar appears in the top part of the Glyph window every time you select the Meter tool . You can use the Glyph window options in the **Application/Preferences** dialog box to prohibit the Meter bar appearing:



Tip: You can temporarily switch to the Meter tool in the Edit mode by pressing and holding down the **CMD+OPTION+SHIFT** keys.

Setting Guidelines, Anchors and Sidebearings

With the Meter tool you can not only measure angles and distances but also mark glyph elements with guidelines and anchors and set right and left sidebearings.

Press the mouse button while holding down the **Cmd** or **Ctrl** key and measure the distance. When you release the button a popup menu appears.

Here is what you can do:

Add two guidelines	Two guidelines, vertical and horizontal, will be added. The point where they will be added depends on the option selected in the secondary menu: 100% means that the guidelines will be added at the end point of the measured distance. 50% means that the guidelines will be added in the middle of the measured line segment and 200% means that guidelines will be added at twice the measured distance.
Add horizontal guideline	The same as above, but only a horizontal guideline will be added. This command with the 50% option can be very useful in finding the middle of a glyph element.
Add vertical guideline	The same, but only a vertical guideline will be added.
Add slanted guideline	A slanted guideline will be added along the meter tool's arrow. Note that the next guideline that you drag from the rulers will be parallel to this one.
Set right sidebearing	Set the right sidebearing at the destination point.
Set left sidebearing	Set the left sidebearing at the destination point. Set sidebearing commands are very useful when you need to set sidebearings at given distance from some glyph element.
Add anchor	An anchor point is added at the destination endpoint of the meter line.

Outline Operations

Operations are temporary tools that let you modify your glyph. Operations are activated by pressing on their button in the Tools toolbar or by selecting a command from the **Operations** submenu of the **Tools** menu or of the Glyph Window's popup menu. There is also the separate Operations toolbar available in the **View/Toolbars** menu:

When an operation is activated one or more handles appear depending on the operation. After you make changes double-click to accept them (you can also press the **Return** or **Enter** key on the keyboard) or press the **Esc** key to reject the changes.

When the operation is completed the tool that was selected will be activated again. As with all permanent tools you can use the zoom selection tool, quick zoom keys and all the other viewing options of the Glyph Window while you are working with the operation tool.

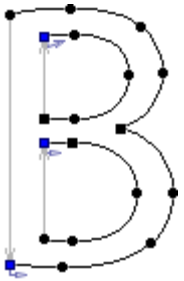
Here is a list of all available operations:

	Transform Selection	Scales, rotates or skews the selected portion of the outline
	Interpolate	Modifies the outline by moving a few "base" points
	Position Background	Sets the size and position of the bitmap background layer
	Envelope	Modifies the outline as if it was painted on rubber
	Curve	Approximates a segment of the outline with a curve
	Set	Lets you set precise positions of outline points
	Reverse Contours	Reverses the contour's direction
	Set Startpoints	Changes the starting points of contours and rearranges contours.

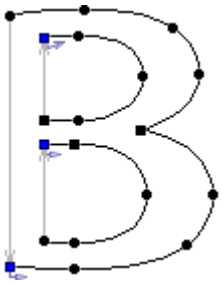
Below you will find a detailed description of the operations that have not already been described.

Interpolation

With this operation you can move a few points and all other points of the glyph outline between the moved points will be interpolated:



Original glyph



Glyph after interpolation

As you can see, this operation can save a lot of time when you want to proportionally modify your glyph but want to save some important values, like stem widths.

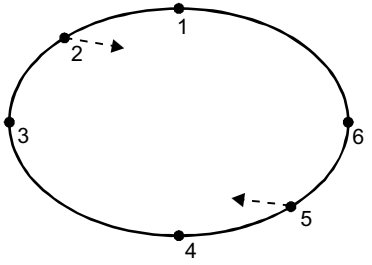
Interpolation is extremely useful when you are making Multiple Master fonts. Refer to the “Multiple Master Fonts” chapter for more information.

When Interpolate operation is activated a small panel appears, consisting only of two buttons, **OK** and **Cancel**:

Press the **OK** button to accept any changes that you made with the interpolation operation (you can also press the **Return** or **Enter** key on the keyboard) or press the **Cancel** button to reject the changes (the **Esc** key is the equivalent of this button).

How interpolation works

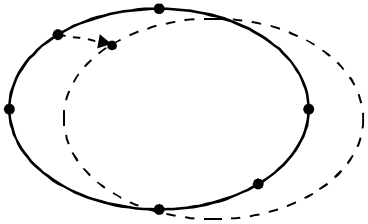
1. You set the new position of several glyph points. We call these points *reference points*.
2. All points on contours that are between two reference points are interpolated. All contours are processed independently:



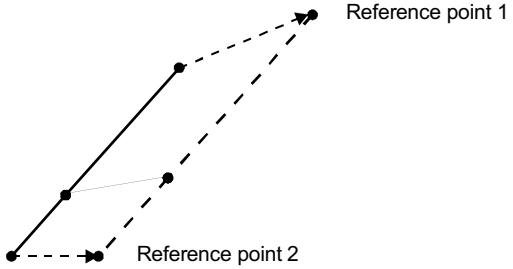
Points 2 and 5 are reference points. Points 3 and 4 are between these points and will be interpolated. Note that points 1 and 6 are *also* between reference points, because all contours in FontLab are closed.

3. All non-reference points are interpolated between two reference points according to the following rules:

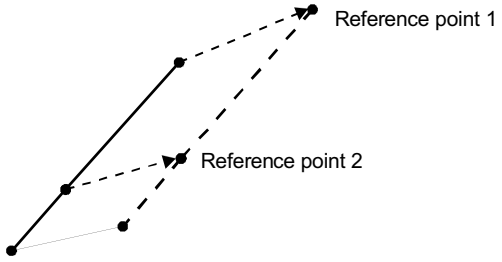
a) If the contour has *only one* reference point, it is shifted with that point:



b) If a point is *between* two reference points, it is proportionally interpolated:



c) If point is outside the interpolated points, it is shifted with the closest reference point:

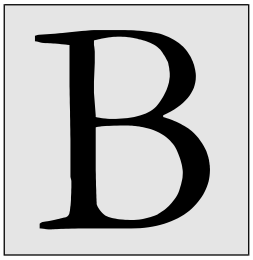


To interpolate a glyph:

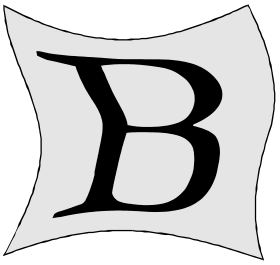
1. Activate an Interpolation operation. Select the **Interpolation** command in the **Operations** submenu of the **Tools** menu or press the button on the Tools or Operations toolbar.
2. You will see a panel with the **OK** and **Cancel** buttons. You can press the **OK** button any time to finish your work with the Interpolation operation or press the **Cancel** button to reject all changes.
3. Position the mouse cursor on the reference point that you want to set; press the mouse button and drag the point to a new location. Release the button when you are done. Note that the point will "stick" to all guiding layers as well as to other glyph nodes. Hold the **Shift** key down to constrain the direction of movement to 15-degree increments.
4. You may want to set a so-called *base point* – a reference point that points to itself. A base point will remain in place during interpolation.
5. When you set the new position of the first reference point, you will see the results of outline interpolation as a gray outline.
6. **Ctrl**-click on a reference point **to remove that reference point**.
7. You can **select several points** using the usual point selection procedure. If you add a reference point after selection all selected points will move to the position of the same reference point. Use this technique to "collapse" parts of the outline.
8. **Ctrl**-click the mouse button in the free space of the Glyph Window and select the **Remove all links command** from the popup menu **to remove all reference links**.

Envelope

This operation transforms your glyph as if it were on a rubber plate and you began to stretch it:



Original glyph

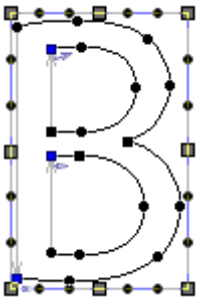


Glyph after envelope transformation

Envelope is the most “freeform” transformation available in FontLab. Be careful with it because it can produce very unusual results.

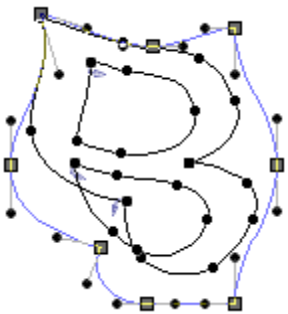
To apply an envelope to a glyph:

1. Activate an Envelope operation. Select the **Envelope** command in the **Operations** submenu of the **Tools** menu or press the button on the Tools or Operations toolbar.
2. You will see a control box surrounding your glyph:



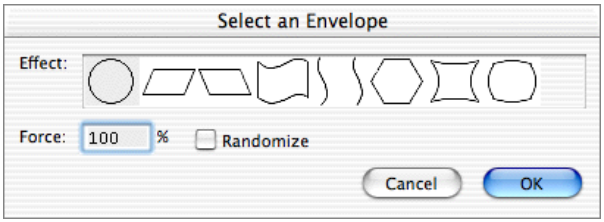
   The control box consists of 8 curves with control vectors and control points.

3. Position the mouse cursor on any of the handles of the control box, press the mouse button and drag the handle to a new location. You will immediately see the results of the transformation:



4. Double-click to accept the changes (you can also press the **Return** or **Enter** key on the keyboard) or press the **Esc** key to reject the changes.

You can select one of the predefined envelopes. **Ctrl**-click the mouse button in the free space of the Glyph Window and choose **Select predefined** command in the popup menu. You will see a dialog box with several predefined envelopes:

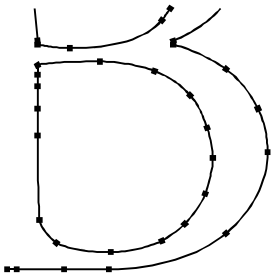


Choose the envelope in the **Effect** list that you want to apply and fill in a **Force** option. A force of 100% will make the envelope look as it is in the selection list. A value of 0% will make a plain rectangle. Check on the **Randomise** option to apply the envelope with random force.

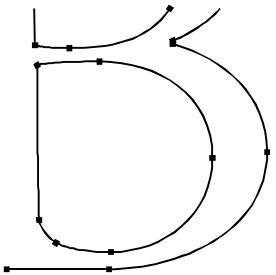
When you select one of the predefined envelopes you can still modify it using envelope handles.

Curve

Using this operation you can approximate a segment of the glyph's outline with a single Type 1 curve (i.e. remove excess nodes in a curve):



Original outline



Outline after manual approximation

To approximate part of the outline:

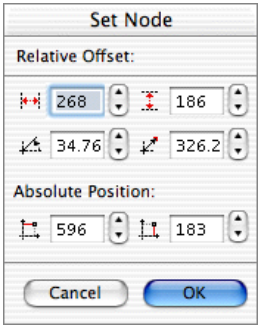
1. Activate the Curve operation. Select **Curve** in the **Operations** submenu of the **Tools** menu or press the button on the Tools or Operations toolbar.
2. Position the mouse cursor on the outline where you want to start the curve. Click the mouse button. Hold the **Shift** key down to make a smooth transition between the new curve and the old outline.
3. Position the mouse cursor on the place on the outline where you want to finish the curve. Click the mouse button. Again, hold the **Shift** key down to make a smooth transition. Note that the starting and finishing points must be on the same contour.
4. Repeat steps 2 and 3 as many times as it takes to approximate the outline. Double-click to accept the changes (you can also press the **Return** or **Enter** key on the keyboard) or press the **Esc** key to reject the changes.

Set

This operation lets you set the position of the nodes very precisely. Of course, you can use the nodes' property panel but this operation may simplify the job:

1. Activate a Set operation by selecting Set in the **Operations** submenu of the **Tools** menu or pressing the button on the Tools or Operations toolbar.
2. Click anywhere in the editing field to set up a reference point. If you click close to any of the guiding elements or close to one of the nodes the reference point will "stick" to that object. The reference point will be highlighted.
3. Position the cursor on the node you want to move and click again.

The destination node will also be highlighted and a dialog box appears:



In this dialog box FontLab shows the relative position of the reference point and the destination node in rectangular and polar coordinates as well as the absolute position of the destination node. You can change any value and all the other values will automatically be recalculated.

After you press the **OK** button the destination node will be moved to the new location and you can repeat steps 1-3 to move other nodes.

Reversing a Contour's Direction

Sometimes you need to reverse the direction of a contour. In FontLab you can do this in one of two ways: **Ctrl**-click on a node of each contour that you want to reverse with the mouse button and select the **Reverse Contour** command in the popup menu, or activate the Reverse Contour operation.

To activate the operation select the **Reverse Contours** command in the **Tools** menu or press the button on the Tools or Operations toolbar.

By double-clicking you can finish the operation and accept all the changes that you just made. By pressing **Esc** you finish the operation and reject all the changes.

➲ **Tip:** If you press **Esc**, but change your mind and want to accept the changes of the last operation, you can use the **Redo** command in the **Edit** menu.

When you activate the **Reverse Contours** operation you will see that all the contours now have arrows that show their direction.

You can reverse any contour by clicking on it with the mouse button.

➲ **Tip:** We recommend that you switch to the **Preview** mode when you use this operation. The direction of a contour changes the contour's "colors," and in the **Preview** (filled) mode you will see the changes immediately.

Rearranging Contours

Sometimes you need to change a contour's sequence to simplify the programming of hint substitution. You can do this by selecting the **Make contour first** command in the node popup menu, which sets contours above each other; but an easier way is to use the **Rearrange Contours (Set Startpoints)** operation.

Select the **Set Startpoints** operation in the **Tools** menu or press the button on the Tools or Operations toolbar.

When you activate this operation you will see the yellow marks that show the sequence number of each contour.

To change a contour's startpoints just click on a new startpoint position.

To change a contour's sequence:

1. Place the mouse cursor on the contour whose sequence you want to change.
2. **Ctrl**-click the mouse button. The selected contour will be highlighted.
3. In the popup menu choose one of these commands:

Up	To move the contour one step up (contour #3 will become #2).
Down	To move the contour one step down (contour #3 will become #4).
First	To move the contour to the top of the sequence (contour #3 will be #1).
Last	To move the contour to the bottom of the sequence (contour #3 will be the last contour in the sequence).

Double-click to accept the changes (you can also press the **Return** or **Enter** key on the keyboard) or press the **Esc** key to reject the changes.

Working with Composite Glyphs

Composite glyphs are glyphs made up of two or more components, like a letter plus an accent. One or more of the components are referenced. I.e. their contours are not actually present in the composite glyph, but are "copied" from and linked to some other character. Thus whenever the original component contour is changed all the composite glyphs that copy the component also change. The contour of composite components appears in dashed lines in the glyph window.

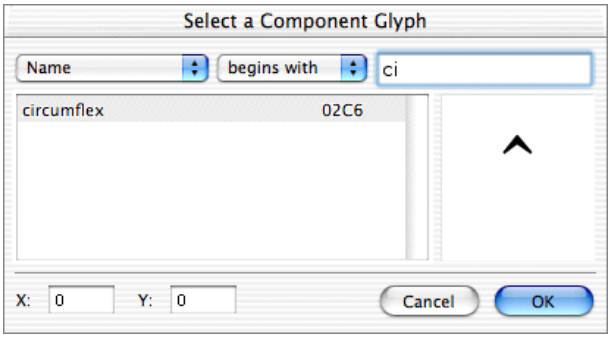
Composites have the advantage of allowing the user to create only one instance of a component that is frequently found in a font and reusing it without having to redraw it each time. Further if the design of the component changes it need only be altered once – in the original component. And finally, a composite takes up less room in the font than an outline, allowing for smaller font files.

There are three operations related to composite glyphs: adding a component to glyphs, decomposing a component and positioning a component.

Adding a Component

To add a component to a glyph currently open in the Glyph Window, select the **Add Component** command from the **Glyph** menu.

You will see a dialog box that looks just like a Get Glyph dialog box:



The difference is that only those glyphs that can be used as component glyphs will appear. For example, a composite glyph cannot be used as a component glyph. And, of course, a glyph cannot be a component for itself.

Another difference is that you can set the position by entering its **x** (horizontal) and **y** (vertical) coordinates. The component position is the distance between the composite zero point and the component's zero point.

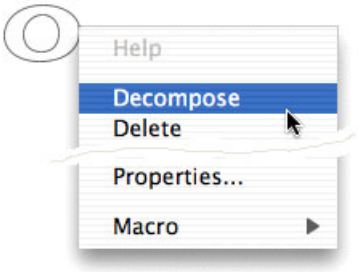
To add a component you select the glyph you want to use as a component in this dialog box and press the **OK** button.

Another way **to add a component** is to drag it from the Font Window and drop it in the Glyph Window while the **Cmd** key is pressed. Make sure you press the **Cmd** key after you started dragging but before you dropped the component.

Decomposing

To decompose a composite glyph select the **Decompose** command from the **Glyph** menu or from the Glyph Window default popup menu. The outlines of all components will be scaled and shifted according to their settings and added to the composite glyph. If the component glyphs had hints then these hints will also be added and a hint replacement program will be automatically generated. The link to the original component will be lost.

To decompose an individual component in a composite glyph **Ctrl**-click the component and select Decompose in the context menu:



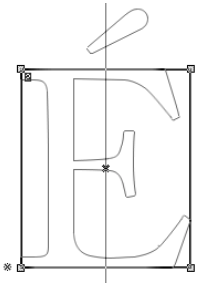
Component Positioning

To activate the component positioning operation, activate the Edit tool, position the mouse cursor on the component's outline and click the mouse button.

Alternately, if the current glyph is composite-only (so if doesn't have any "normal" outlines), use the **Page Up** and **Page Down** keys **to select a component for editing**.

You will see a control box surrounding the component with four corner handles, a cross in the center, a centerline and the number of the component in the components list.

To select another component, use the **Page Up** and **Page Down** keys or the **Tab** key.



To move the component position the mouse cursor inside the control box, press the mouse button and drag the control box to a new location. If you position the cursor on the cross in the middle of the control box you can set the position of the component more precisely because the cross will snap to the guiding elements while moving. Pressing the **Shift** key will constrain the movement in horizontal or vertical directions, pressing **Cmd-Shift** allows the component to move only horizontally.

When you move the component, it will "stick" to its original position. You must hold down the **Option** key to avoid this.

You can also **use the keyboard to move the component**. Arrow keys move the component in one font unit increments and the **Shift**+arrow keys increase movement to 10 units.

To scale a component position the mouse cursor on one of the handles, press the mouse button and drag the mouse to change the size of the component. Hold the **Shift** key down to constrain the proportions of the component.

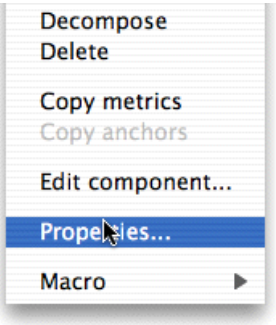
Just click outside the component's control box to accept the changes (you can also press the **Return** or **Enter** key on the keyboard) or press the **Esc** key to reject the changes. You can also click another component to select it for editing.

Some other useful commands are available in the popup menu that appears if you **Ctrl**-click the editing area while component tool is active:

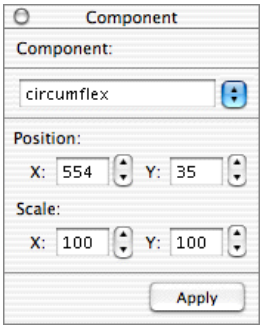
Decompose	Decomposes (adds the outline to the composite glyph) the current component.
Delete	Removes the component.
Reset Shift	Moves the component to the position (0, 0).
Set Scale 100%	Sets the scale at 100% for the component. Note that the Type 1 font format does not support scaled components. The Scale 100% button in the control panel of this operation has the same effect.
H Mirror **V Mirror**	Mirrors the component (sets the scale for x or y direction to −100% and adjusts shift accordingly).
Make First	Sets the component to the first place in the components list.
Copy Metrics	Copies metrics data from the component to the composite glyph.
Copy Anchors	Copies all anchors from the component glyph to the composite.
Edit Component	Opens a new Glyph Window with the currently active component.
Properties	Opens the Component Properties panel (described later).

Component Properties

You can **set the precise size and position of the component**. Ctrl-click the component with the Edit tool. You will see a popup menu:



Select the **Properties** command in this menu and you will see the component Properties dialog box:



In this dialog box you can select a different glyph to be used as a component and set the component's position and scale. The component position is the distance between the composite zero point and the component's zero point.

➲ **Tip**: You may just double-click on the component to get the component Properties dialog box.

Using the Primitives Panel

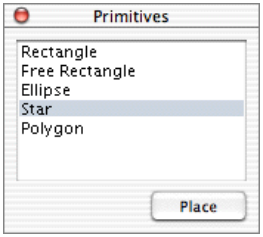
In FontLab you can add predefined customizable graphical primitives to glyphs. There are five primitives included with FontLab and it is possible to add more. You can select a primitive from the Primitives panel.

To open the Primitives panel select the **Primitives Panel** command in the **Windows** menu.

The panel consists of a primitives list and a **Place** button that you click to add a primitive to the outline. Here is a brief description of each primitive:

Rectangle	Simple rectangular contour
Ellipse	Circle or ellipse
Free rectangle	Rectangle that can be freely rotated
Star	Star with customizable number of rays
Polygon	Polygon with customizable number of vertices

To add a primitive to a glyph's outline, select the primitive that you want to add and press the **Place** button on the Primitives panel. Or just double-click the primitive's name in the primitives list:



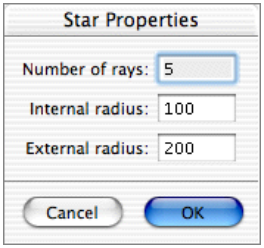
When you press the **Place** button you will see the primitive appear in the middle of the Glyph Window. Primitives have a blue outline and several control handles that you drag to customize their shape.

Drag the cross control handle in the center of the primitive **to move it**. The behaviour of the other handles depends on the primitive's type. For example, in the Star primitive the handles determine the internal and external radius of the star's rays and the angle between vertices that form a ray.

Try placing some primitives into the Glyph Window and drag their control handles to see what they do. Use the **Shift** key to constrain the movement of handles. For example, in the Rectangle primitive holding the **Shift** key down will produce a square instead of a rectangle.

Every primitive has a property panel where you can set the primitive's parameters precisely. Press the ... button on the control panel to open the Properties panel for the primitive:

Here is a preview of the Star's properties panel:



In this panel you can change the number of the star's rays and the values of the internal and external radii.

FontLab has a standard for primitives modules, so if you visit our Web site you may find more primitives as add-on modules.

Importing and Exporting Glyphs

The most common format for contour-based data is Encapsulated PostScript (EPS). Because this format is a native format of a well-known contour-manipulation program, Adobe Illustrator, another name for it is Adobe Illustrator format (AI). If we go deeper, we see that EPS is much more common than AI, but due to a well-defined internal structure AI format has become a standard way of exchanging contour-based graphics.

With FontLab you can export any glyph as an AI-compatible file and use it in any other program and you can import any EPS/AI file (compatible with Adobe Illustrator) to add to any glyph of your font.

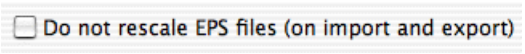
To export a glyph as an EPS/AI file:

1. Select the **Export to EPS** command from the **Glyph** menu.
2. Select the export folder and enter the name of the EPS/AI file in the standard File Save dialog box.
3. Press the **Save** button in the dialog box, and the EPS/AI file will be exported to the folder chosen.

To add an EPS/AI file to the current glyph:

1. Select the **Import from EPS** command from the **Glyph** menu.
2. Choose the file that you want to import in the standard File Open dialog box.
3. Press the **Open** button in the File Open dialog box, and the EPS/AI file will be imported, added to the current glyph, and selected to simplify its modifications.

When you import or paste an outline from another application into FontLab it scales it to fit the font's UPM. To prevent an outline from scaling, use the **Do not rescale EPS files** option located in the **Preferences/General** dialog box:



You can also **export several glyphs at once**: switch to the Font Window (for example, by selecting the **Open Font Window** command from the Glyph Window's popup menu), select the glyphs that you want to export and select the **Export to EPS** command from the **Glyph** menu. You will see a Save File dialog box where you enter a prefix file name for the exported glyphs. Each glyph will be exported to its own file with the file name consisting of the prefix plus the sequential number of the exported glyph.

Printing a Glyph

To print a sample of the current glyph select the **Print** command in the **File** menu while the Glyph Window is active.

You will see a standard Print dialog where you can choose a printer and select other printing options.

To begin printing click the **Print** button. When printing is complete you will have a sample of the current glyph as seen in the Glyph Window. If you see nodes on the screen they will appear on the paper. If you are in preview mode the printed glyph sample will be filled.

Zoom mode on the screen will also be applied to the printed sample of the glyph. Using this feature you can print very detailed samples of the glyph.

Editing Font Metrics

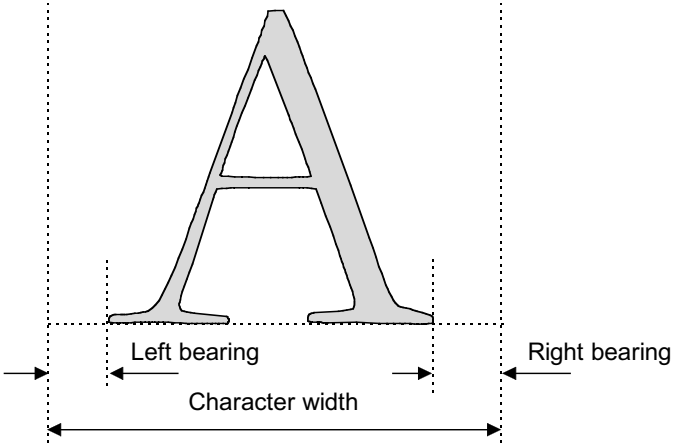
The tools in FontLab that you can use to edit metrics data are common to all FontLab applications, so if you have learned how to use these tools in FontLab you will be ready to use these same tools in any of the FontLab programs.

5

What are Font Metrics?

A program that aligns and spaces text calculates the total width of all the characters in a paragraph. It then adjusts the widths of the space characters that separate the words and tries to put as many characters as possible into one line. The information about the words that are used to make a paragraph, and the information about the width of the individual characters is the only information necessary. This is what is known as font metrics.

To position the individual characters, the page layout program must have information about each character's sidebearings:



Using this information the page layout program positions all the characters. Kerning information is also necessary in order to get the best results.

Kerning

Kerning information is used to adjust the space between specific pairs of characters. As you can see in the following picture some characters may be well spaced with just the bearings rectangle but other characters are not. To fix this problem a special technique called kerning has been developed.

Kerning defines the modification of inter-character space in specific pairs. A good example is the AV pair. In the following picture you can see two examples of inter-character spacing, with and without kerning:

With kerning *Without kerning*

You can see that only the kerned image is optically correct because it can compensate for the problem caused by the special form of the V and A characters printed in sequence leaving too much space between the letters.

Metrics Files

Information about a glyph's width is usually located in font files. Kerning information may also be included in the file. In TrueType and FontLab font formats both metrics and kerning data are located in a single font file. In PC Type 1 (PostScript) fonts the metrics and kerning data are located in separate files.

There are two possible formats for the metrics files that are used with PC Type 1 fonts: AFM and PFM. AFM files (*Adobe Font Metrics*) are text files containing all the metrics and kerning information for a given font. These files are legible as text and can be edited in any text editor. PFM files (*Printer Font Metrics*) are metrics and kerning files used by the Windows operating system. They are binary files and cannot be read without special utilities. AFM files are a standard format for the exchange of metrics information for PostScript fonts. This information can be read directly by several operating systems and programs.

In the Windows OS Multiple Master Type 1 fonts store metrics information in another format. There is a binary Multiple Master Metrics file, which usually has the file extension MMM and a text Multiple Master metrics file. MMM files include information about the font's axes and masters, multiple-master data for glyphs' metrics and kerning and other font header data. The text metrics files have a composite structure: there is a linking file that describes the font header, axes and masters information, and separate AFM files (with metrics and kerning data) for each master font.

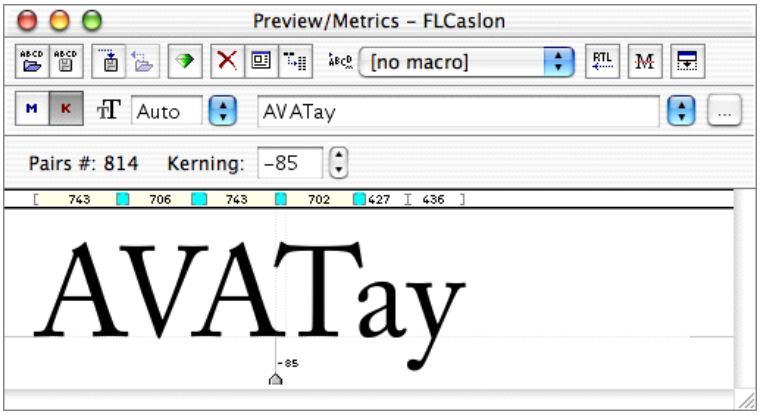
On the Macintosh information about Multiple Master metrics is stored in the font suitcase data structure and is handled automatically by the Type Manager.

FontLab can import and export metrics and kerning information in any of these formats.

Metrics Window

FontLab has a special window where you can edit the metrics and kerning information. It is called the Metrics Window.

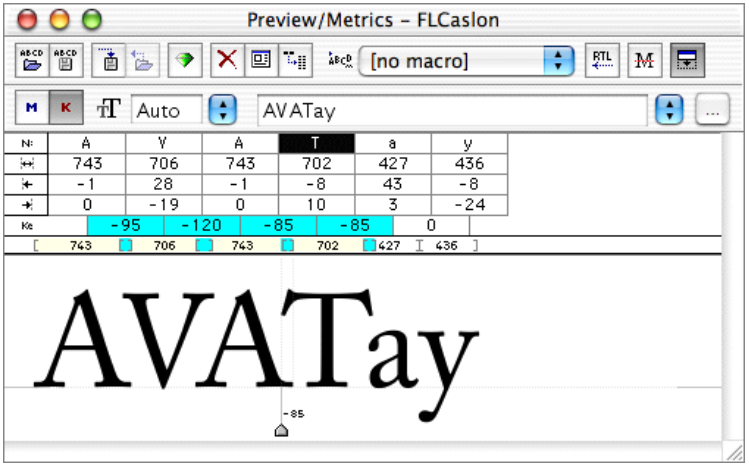
To open the Metrics Window select the **New Metrics Window** command in the **Window** menu. The Metrics Window will appear:



The Metrics Window consists of several parts:

- A toolbar with controls for importing and exporting metrics files, automating metrics or kerning generation and other commands.
- An area that is used to select a mode for the Metrics Window and a string for metrics or kerning editing.
- The property area includes “live” digital information about metrics or kerning for the current glyph or glyph pair.
- The editing area where the edited string with controls appears.

The property area of the Metrics window can work in two modes: compact and expanded. In expanded (default) mode information about glyph metrics is previewed in a table located at the top of the editing area:



To switch from compact to expanded mode, use the **Expand** button: .

There are three possible modes of the Metrics Window: **Preview**, **Metrics** and **Kerning**. The default is the **Metrics mode**.

Use these two buttons to change the mode of the window. Press the **M** button to switch to the metrics mode (glyph metrics are editable but kerning is not), **K** button – to the kerning mode (kerning is visible and editable) and release both buttons to return the window to the Preview mode.

In the **Preview** mode you cannot edit the metrics or kerning data of the font. You can only preview the sample string and look at the glyphs' alignment.

In **Preview** if the properties area is in compact mode you can edit the position of the underline line that will appear if the font that you edit is used in the underline mode. Of course, you can also edit the Underline position and thickness in the Font Info dialog box on the Basic dimensions page.

To change editing modes you can also **Ctrl**-click the editing area and select a mode for the Metrics window in the popup menu.

Metrics Window Toolbar

This is a simple list of all buttons available on the toolbar:

	Opens a metrics file (PFM, AFM or MMM format)
	Saves a metrics file
	Quick Save – saves the current metrics files to a temporary file
	Opens the metrics from a temporary file saved by Quick Save
	Opens the Automatic Metrics or Automatic Kerning generation dialog boxes.
	Opens the Reset Kerning dialog box
	Opens the Kerning Editor dialog box
	Allows selection of special macro programs that can be used to simulate complex environments to test metrics and kerning
	Opens the OpenType Kerning Assistance dialog box
	Changes the preview panel to the right-to-left reading mode
	Activates the support for the measurement line. When it is active, all metrics are measured along the line
	Alternates between compact and expanded mode of the editing area.

Selecting a String for Previewing or Metrics Editing

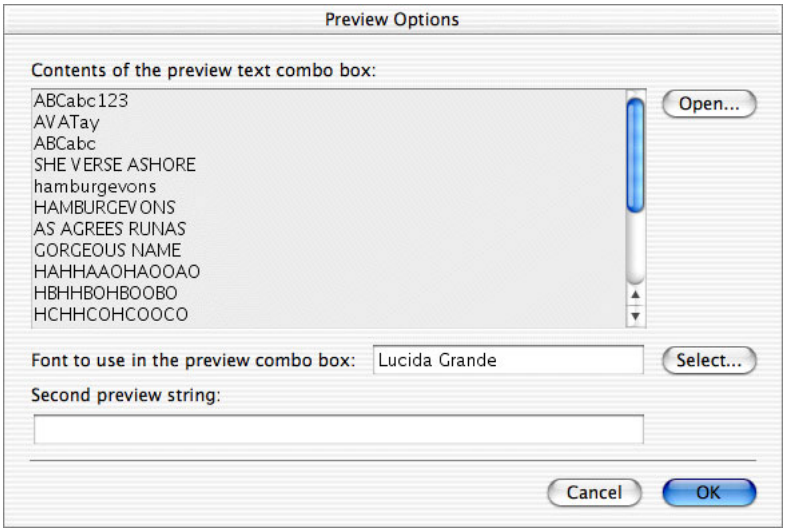
If you are in the **Preview** mode you may want to enter a string to preview. The easiest way to do this is to enter the string in the editing field of the Metrics Window and press **Enter**:

Or, you can open a list of predefined strings and select one of them for editing:

By pressing the **Cmd-PgDn** and **Cmd-PgUp** keyboard shortcuts you can browse the strings quickly.

All strings that appear in the list box are read from the Preview.txt file located in FontLab/Data folder. This file is an ordinary text file and can be edited in any text editor to add your own strings.

You can edit the contents of the strings list in the Preview Options dialog box. **To open the dialog box**, click on the ... button:



You will find an editing field containing all the strings that appear in the list box. Edit the text or use the **Open...** button to fill it from an external text file.

If you are working with a non-Latin font you may want to select the font that is used in the sample string control and list. Use the editing field below the contents field to enter the desired font name or click on the **Select...** button to select the font with the standard font selection dialog box.

The bottom position in the dialog box is occupied by the “Second preview string” control. Text that you enter here will appear as the second line in the Metrics window:

Note that this second line is not editable and is used to make visual comparison to the editing metrics of other glyphs in the font.

Navigating in the Sample String

Use the and buttons on the Tools toolbar **to move to the previous or next glyphs in the string**. These buttons are useful if one of the glyphs has a negative width value.

You can also use the **Page Up** and **Page Down** keys on the keyboard to navigate in the sample string.

To scroll the window you can press the space bar and scroll the Metrics window with the hand tool.

Entering Special Characters

Some glyphs are not represented by the ANSI characters. However, by using special rules you can enter non-ANSI names in the sample string.

To enter a glyph's name, begin with the slash character (/) and finish with a space character or another slash: `/A.small/T.small`

To enter a slash, type it twice (//).

The decimal code of a glyph can follow the slash instead of using its name: `/44`

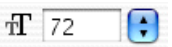
The Unicode index of the glyph can be entered if you use the "\" prefix: `\0445\0448`

To enter a backslash type it twice: "\\".

The easiest way to fill a sample string is **using the drag-drop method**. You can simply drag any glyph from the Font Window and drop it in the Metrics Window and it will be added to the sample string. If you hold the **Cmd** key while dropping the selected glyphs, they will replace the sample string.

Selecting Previewing Size

In the **String Size** combo box:

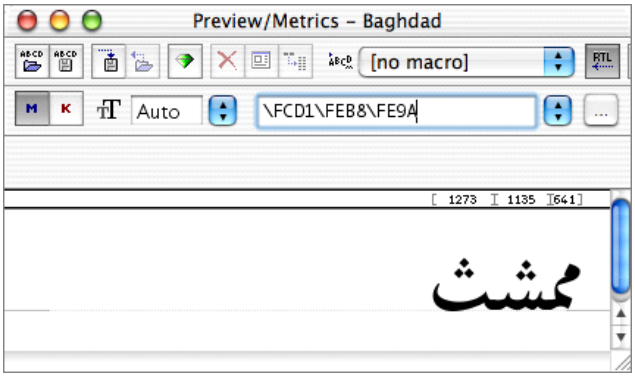


you can select the size of the previewed string. By default, "Auto" is selected and the size of the previewed text is calculated dynamically, depending on the height of the window.

If the string becomes too large to fit in the window, a vertical scroll bar will appear allowing you to view all the editing areas of the Metrics Window.

Right-to-Left Mode

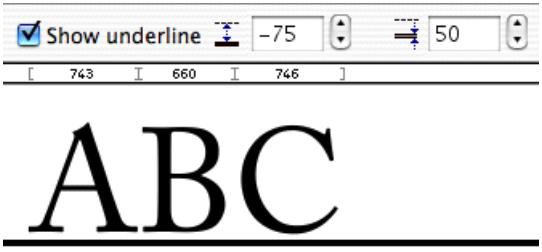
If you are working on a font that requires right-to-left reading, like Arabic or Hebrew, you can change the Metrics window to the right-to-left mode. Click on the RTL button on the toolbar and you will see that the preview string is written from right to left:



Editing an Underline

In the **Preview** mode of the Metrics Window you can view and edit the position of the underline that appears in a text editing application when the underline style is chosen. The position and thickness of an underline are considered a font parameter and are part of the font's design.

To view an underline in the Metrics Window **Preview** mode, switch on the **Show Underline** check box located in the property area. Be sure that the properties area is in compact mode (so the properties line is visible):



You will see the underline appear in the editing area and the underline controls in the property area of the Metrics Window:

The left control is used to edit an underline's position and the right control the underline's thickness. If you change values in the control boxes you will see the underline's position and/or thickness modified.

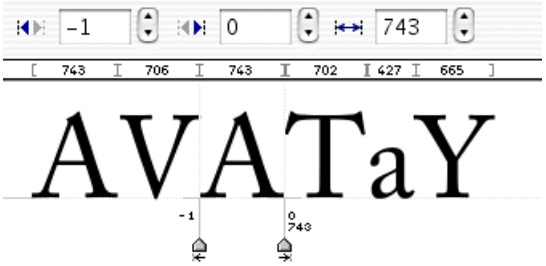
Editing Metrics

By “metrics”, we refer to information about a glyph’s width and sidebearings. In FontLab you can modify this information either manually or automatically.

To modify a glyph’s metrics you must switch the Metrics Window to the Metrics mode. To do this, push the M button in the mode selection area:

or, select the **Metrics** command in the popup menu that appears if you **Ctrl**-click in the editing area of the Metrics Window.

By default the Property area is empty. To make the metrics editing controls visible, click on a glyph in the editing field. The metrics editing controls will appear and the sidebearings lines with editing handles will appear at the sides of the glyph:



The numbers at the bottom of the glyph are the left and right sidebearing values and the glyph’s width.

Manual Metrics Editing

To modify a glyph's metrics you can use three methods:

1. Move the sidebearings lines using the mouse.
2. Edit the values in the property area of the Metrics Window.
3. Drag the glyph within the editing area.

To move the sidebearings lines just position the mouse cursor on the line, press the mouse button and drag the mouse. Release the mouse button when you are done.

To drag a glyph within the editing area, position the mouse cursor on the glyphs' image; press the mouse button and drag the mouse to position the glyph inside its width. Press the **Option** key while dragging the mouse to modify the glyph's width.

You can easily **modify the vertical position of the glyph** relative to its baseline. Just press and hold the **Shift** key on the keyboard while dragging the glyph and this will modify its vertical position.

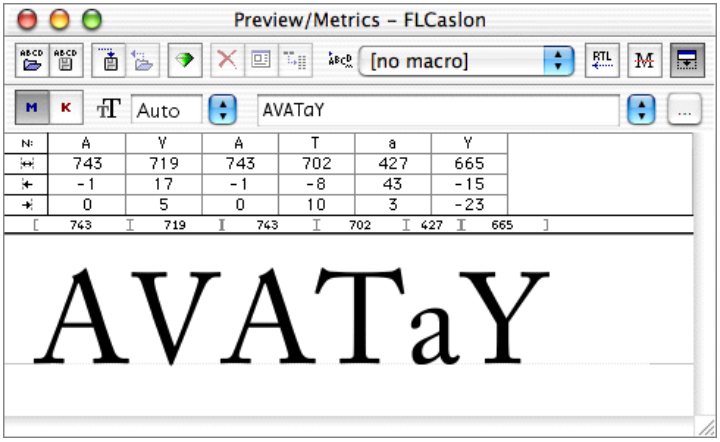
Using the Keyboard

When the glyph is active, you can use the keyboard to adjust the metrics:

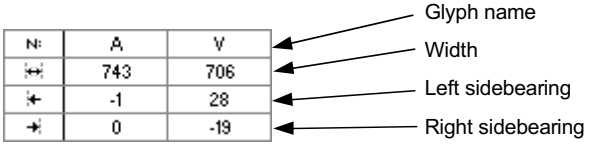
Left and right arrow keys	Moves the glyph by one font unit inside the sidebearings without changing the advance width. Hold the SHIFT key to move the glyph by 10 font units.
Cmd+left and right arrow keys	Moves the glyph together with the right sidebearing. This changes the left sidebearing and the advance width. Hold the SHIFT key to move by 10 font units at each key click.
Insert	Changes the current glyph to the next glyph in the font.
Delete	Changes the current glyph to the previous glyph in the font.
Page Up	Moves to the previous glyph in the sample line.
Page Down	Moves to the next glyph in the sample line.
Any character or digit	Selects the character you have clicked as the current character for editing. You can also enter the glyph name if you want to access glyphs that are not assigned to any key combination.

Using the Metrics Table

Click on the button to switch the properties area to expanded mode. You will see the metrics table appear in the top area of the editing field. The properties line will disappear:



In Metrics mode the properties area table consists of four lines:

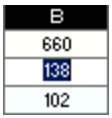


Click on any number in the table to enter an exact value. Just use the keyboard to adjust the number and press the **Enter** key when you are done. The **Esc** key or a left click outside the cell you are editing will cancel the changes.

Use the **up** and **down** arrow keys on the keyboard to move up and down in the table. Use **Tab** key to move right and **Shift+Tab** to move left.

Referencing Metrics

In the table you can use a reference instead of the real numeric value. For example, if you want to set the left sidebearing of the glyph ‘B’ to be equal to the left sidebearing of the glyph ‘D’, click on the cell located at the intersection of the B column and the third row:

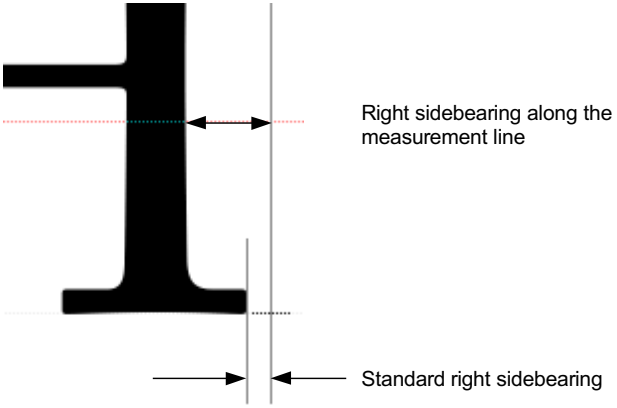


and, instead of the numeric value for the metric, enter “=D”. When you click **Enter** key to accept changes, the data will be copied from the source glyph.

The Measurement Line

To activate the measurement line, click on this button: . You will see a red line appear in the editing area:

When the measurement line is active (and visible), the glyph sidebearings are measured not from the extreme points of the character, but from the point of intersection of the measurement line and the contour:



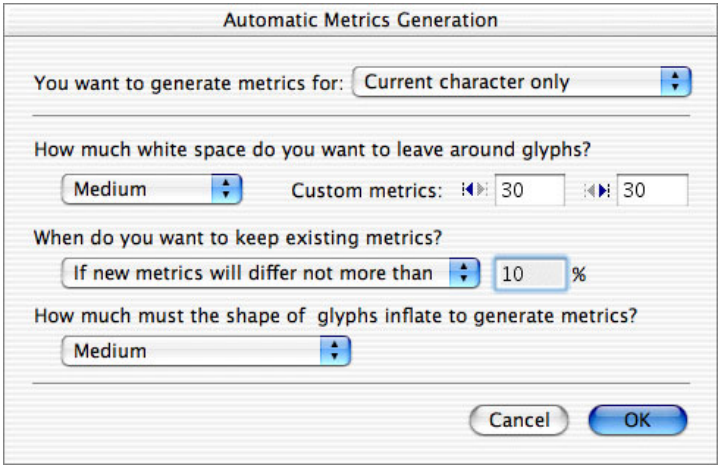
The measurement line is very useful when you are setting metrics in a serif font – usually serifs are ignored when sidebearings are calculated and the measurement line gives you a natural way to do that.

Automatic Metrics Generation

FontLab can automatically define a glyph's metrics using a special algorithm. This algorithm usually produces good results but we recommended manual editing for the best results.

To define the metrics parameters automatically, click on the button on the toolbar or select the **Auto Metrics** command in the context popup menu.

The **Automatic Metrics Generation** dialog box appears:



This dialog box includes two areas: **Area of application** and **Parameters**. In the first area you select the glyph(s) to which the automatic algorithm will be applied.

The possible choices are:

Current character only	This option is available and is the default if any glyph is selected in the editing area.
All glyphs in the current string	This option generates metrics for all glyphs in the current string in the editing area.
Whole font	This forces FontLab to generate metrics for all glyphs in the font and is not generally recommended. This operation is not undoable. FontLab prompts you and asks that you save the current metrics information in a special file from which it may be easily restored if you are not satisfied with the results that the autometrics algorithm produced.

You can choose the parameters for the algorithm in the **Parameters** area of the Autometrics dialog box. All the parameters are displayed. We recommend that you experiment with various parameters using the autometrics application.

Quick Save and Quick Open

You can use these commands to temporarily save the current state of the metrics and kerning information. To quick save a metrics file, ~SAVE.AFM in the FontLab folder, press the button on the toolbar.

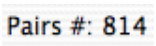
To open a previously saved file, press the button. The Warning dialog box appears prompting you to save the current (modified) state of metrics into the same temporary file.

Editing Kerning

To edit kerning data switch the Metrics Window to Kerning mode by pushing the K button in the mode selection area:

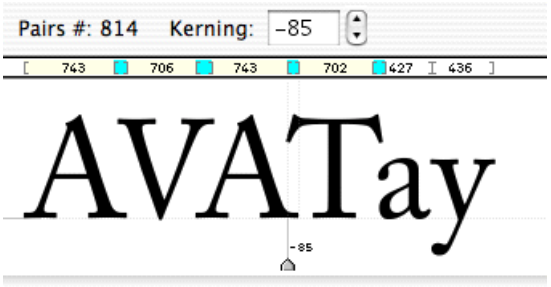
or, select the **Kerning** command in the popup menu that appears if you CTRL-click in the editing area of the Metrics Window.

When you switch to the kerning mode you will see the total number of defined kerning pairs for the current font appear in the property area of the Metrics Window:



To make the Kerning Editing controls visible you must select the pair that you want to edit. Position the mouse cursor on the right glyph of the pair in the editing string and click the mouse button.

You will see the Kerning Editing control appears in the property area and the kerning line and handle appear in the editing area:



There is now a blue area in the metrics bar at the top part of the editing area. This means that kerning exists for that pair in the current preview string.

Manual Kerning Editing

To edit kerning manually you must drag the kerning line (or right glyph of the kerning pair) using the mouse. If you **Ctrl**-click the mouse button while dragging the glyph **kerning for that pair will be removed**. You will see that the total number of kerning pairs decreases.

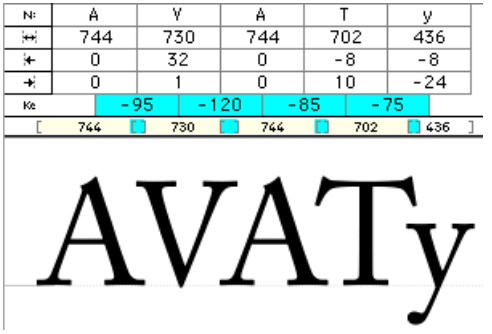
Using the Keyboard

When a glyph is selected in the sample string you can use the left and right arrow keys to change the kerning by one font unit at each key click. Hold the **Shift** key to change the kerning by 10 font units.

Use the **Insert** and **Delete** keys to change the glyph in the string and **Page Up** and **Page Down** keys to move to the previous and next glyph in the string.

Using the Table

When the properties area is expanded and kerning editing mode is activated you can see all the glyph metrics and pair kerning information in the editing field:



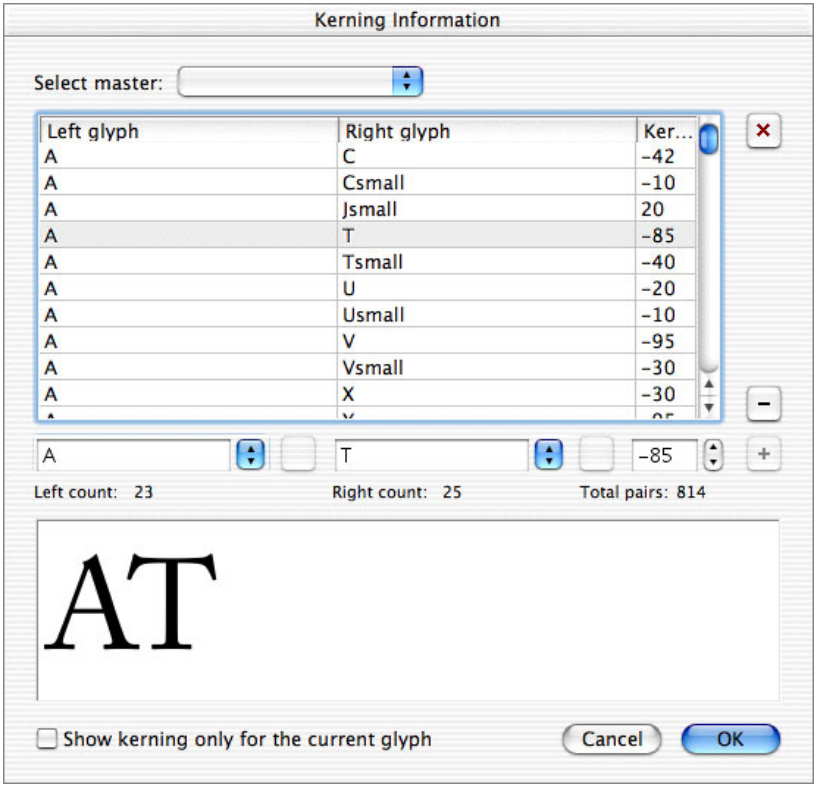
Kerning is displayed on the fifth row in the table and each value is positioned between the glyphs that form the kerning pair. Background color for the kerning value is white when there is no kerning, light blue if kerning is negative (glyphs are shifted toward each other) and yellow if kerning is positive.

To change the kerning value, click on the kerning row in the table and enter the new value. Click **Enter** key on the keyboard to accept the changes or **Esc** to cancel. Use the **Tab** and **Shift+Tab** keys to select a pair in the sample string.

Using the Kerning Dialog

You may add or delete kerning pairs linked to a selected glyph and enter precise kerning values using the Kerning Information dialog box.

To open this dialog box press the button on the Kerning window toolbar or select the **Edit Kerning** command in the popup menu. You will see the following dialog box:



In the top half of the dialog box you see a table that has information about all the kerning pairs in which the current glyph is on the left. Each row of this table includes the name of the paired glyph on the right and the kerning value.

You can sort this list by the names of the “right” glyphs or by the kerning value. **To sort the list**, click on the header of the column that you want to use as the sort key.

You can **edit a kerning value** or **change the right glyph of the pair** using the edit controls below the list. The sample window in the bottom part of the dialog box previews the selected kerning pair. You will immediately see the result of the changes in the Metrics window.

To add a new kerning pair press the **Add** + button. The currently selected pair will be duplicated and you can change it. This is very useful when you want to add a kerning pair that has a value equal to an existing pair but the right glyph is different, as in the “Ta” and “Tà” pairs.

To remove a kerning pair select it in the list and press the **Del** – button.

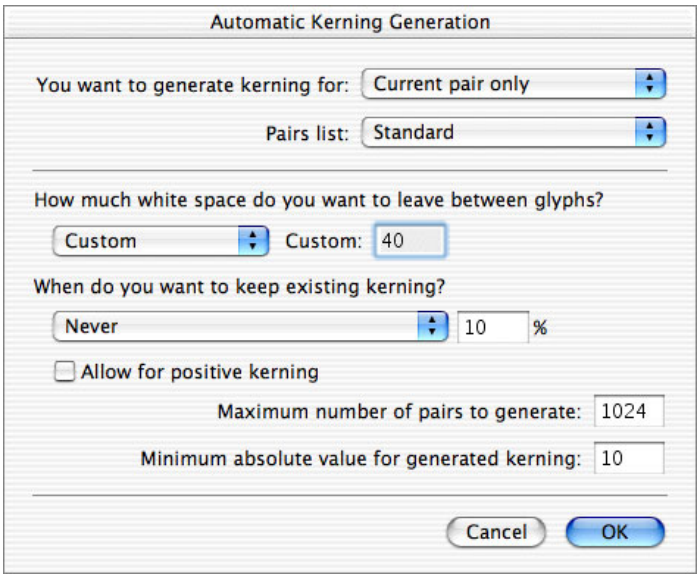
You may also remove all kerning pairs associated with the current glyph by pressing the **Reset** × button.

Automatic Kerning Generation

The easiest way to apply kerning to a font is to use FontLab's auto-kerning algorithm. This algorithm analyses the shape of the glyphs in the given pairs and automatically kerns them. You can control the pairs list that the autokerning algorithm processes as well as other parameters.

To define kerning automatically press the button on the toolbar, or select the **Auto Kerning** command in the popup menu.

The **Automatic Kerning Generation** dialog box appears:



This dialog box consists of two areas: the **Area of application** and **Parameters**.

In the first area you select the pairs for which the algorithm will compute kerning values. You can choose between **Current pair only** (available if one of the pairs is selected in the editing area), **All Pairs in the current string,** or **All Pairs in the following list**.

The second popup menu allows you to generate kerning for all the pairs located in a special list file. The list files are stored in the FontLab/Kerning folder. You can create your own kerning pair files or use files placed there by FontLab's installer.

The **Parameters** area lets you customize the autokerning algorithm. The most used option is: **How much white space do you want to leave between glyphs?** This controls how close the glyphs will be moved together while computing kerning in the pair.

The **Allow for positive kerning** check box lets the autokerning algorithm produce positive kerning in pairs. Positive kerning moves glyphs apart from each other. Positive kerning is usually not recommended but there may be occasional circumstances where it is needed.

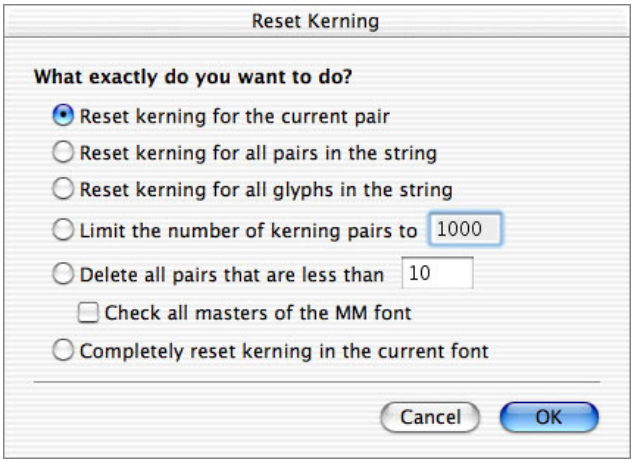
If you want to save the existing kerning the popup menu lets you control the disposition of the existing (imported or manually created) kerning pairs. You can replace existing pairs by automatically generating new ones, keep them unchanged, or select the condition mode.

The **Maximum number of generated pairs** and **Maximum absolute value for generated kerning** options control the possible number of automatically created pairs and the maximum normal (negative or positive) kerning value.

Resetting Kerning

To remove the kerning information for some glyphs or for the entire font you must use the *Reset Kerning* feature. To open the Reset Kerning dialog box press on the or select the **Reset Kerning** command in the popup menu.

The **Reset Kerning** dialog box appears:


Reset Kerning
What exactly do you want to do?
Reset kerning for the current pair
Reset kerning for all pairs in the string
Reset kerning for all glyphs in the string
Limit the number of kerning pairs to
1000
Delete all pairs that are less than
10
Check all masters of the MM font
Completely reset kerning in the current font
Cancel
OK

This dialog box includes options that control kerning removal.

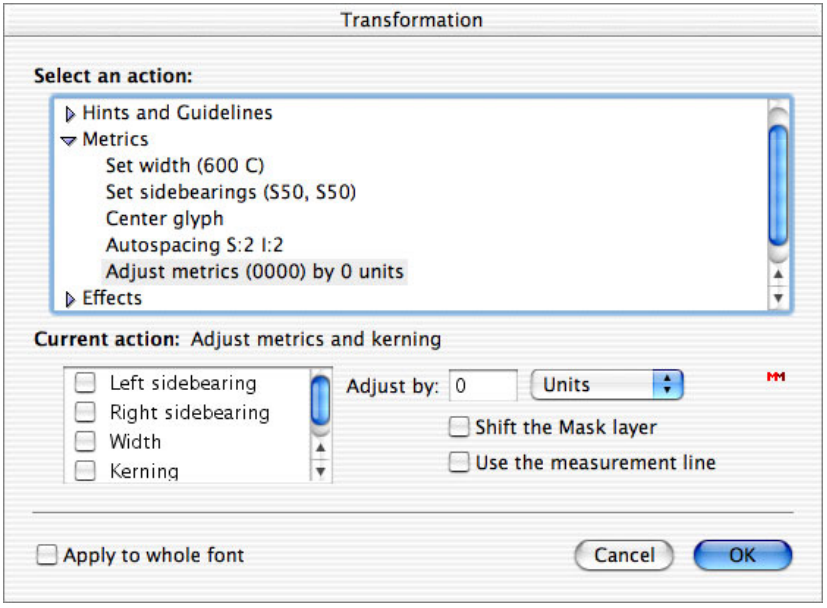
Available options are:

Reset kerning for the current pair	This is the default if a pair is selected. Removes kerning for that pair only. You can get the same result by Ctrl-clicking while editing kerning in the current pair.
Reset kerning for all pairs in the string	Default if no pairs are selected. Removes kerning in all pairs that exist in the current string.
Reset kerning for all glyphs in the string	Removes kerning in all pairs that include glyphs in the current string.
Limit number of kerning pairs to...	Kerns only the given number of pairs with the largest absolute kerning value.
Delete all pairs that are less than...	Removes all pairs that have a kerning value less than the given value. The absolute value of kerning is compared.
Completely reset kerning in the current font	Removes all the kerning pairs available in the font. Because this is not an undoable operation, the warning dialog asks you to save the current metrics and kerning data in the temporary file.

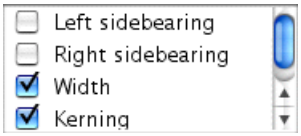
Adjusting Metrics and Kerning

If you want to change metrics or kerning by some fixed value and the manual process will take too much time, you can use the Transform dialog box.

1. Switch to the Font window.
2. Select the glyphs that you want to process.
3. Open the Transform dialog box using the **Tools/Transform** command. (Do not confuse this with the Transform Selection operation.)
4. In the dialog box select the Metrics/Adjust metrics action:



5. In the options area select the metrics that you want to change:



6. Enter the amount by which you want to adjust the values and select the units (which can be font units or a percentage of the source value). Enter a positive number to increase metrics distances or kerning or a negative value – to decrease.

   Normally metrics are measured from the bounding box. You can change this, however, if you want, to measurement from a measurement line by using the check box.

   The measurement line is a horizontal line (defined in the Glyph Window) that defines a "slice" which is used to measure the distance from outline to sidebearings. Please refer to the page 307 in the "Glyph Window" chapter for more detail.

7. Press the **OK** button to transform the selected glyphs.

Class-Based Kerning

Some glyphs in a font may have a similar shape and may be kerned equally. For example, “A” and “Acaron” will have the same kerning value if kerned with “T” or “Tcaron”. With the standard kerning system this will require 4 kerning pairs. If the number of similar glyphs in the group increases, the number of necessary pairs will rise dramatically.

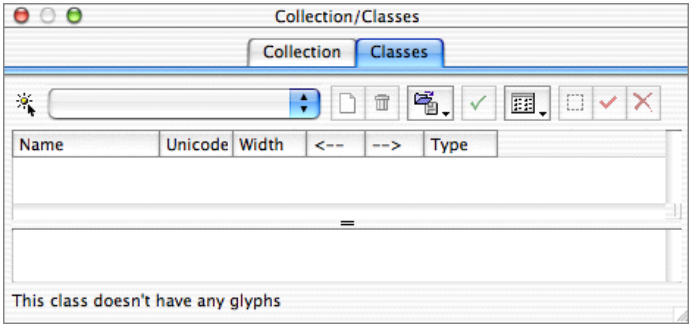
A better way to handle the kerning of similar glyphs is to define glyph groups or classes. In our example the first class will contain the glyphs “A” and “Acaron” and the second “T” and “Tcaron”. Then we will need to define only one pair and all glyphs in both classes will be covered.

Kerning classes may save a lot of time if you need to kern a font that contains a lot of similar glyphs. The typical number of class kerning pairs (which define kerning between classes) is a few hundred. Compare this to several thousand pairs that are necessary to define the same kerning data using traditional methods.

Classes are also actively used to define OpenType features that are applicable to a set of glyphs, so we will return to this subject in the “OpenType Fonts” chapter.

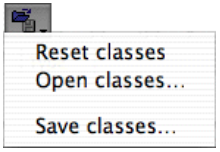
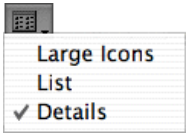
Classes Panel

To define glyph classes in FontLab you need to use the Collection/Classes panel. Open the panel using the **Collection/Classes panel** command from the **Window** menu. You will see a panel with two pages: Collection and Classes. Activate the Classes panel:



This panel contains a toolbar, a class preview panel, a class definition panel and a status bar that tells you how many glyphs are defined for the current class.

On the toolbar there is a list of currently available classes and several buttons:

	Add class	Defines a new class
	Remove class	Removes the current class
	Accept changes	Click the button to accept the changes made to the class definition code
	File operations	 **Reset classes** – removes all classes **Open classes** – opens the class information from the text file **Save classes** – saves information about the font classes to a text file.
	Choose view mode	Use this button to select one of the class preview modes in the drop-down menu: 
	Select class	Selects all glyphs that belong to the current class in the Font window
	Mark class	Marks all class glyphs in red
	Remove glyph	Removes the glyphs selected in the class preview panel from the class

Class Definition

Every class is defined as a list of glyph names preceded by the class name:

```
_kern105: scircumflex' scaron
```

Kerning class names must begin with the underscore glyph – this is necessary to distinguish them from the "plain" classes used in OpenType. The class definition appears in the bottom part of the panel and you can edit it to add or remove glyphs. After you make changes, click the **Enter** key on the keyboard or the button and the preview panel will show the list of glyphs that are named in the class and can be found in the current font.

Defining the Class

Click on the button **to define a new class**. You will see a default class name appear in the class definition panel:

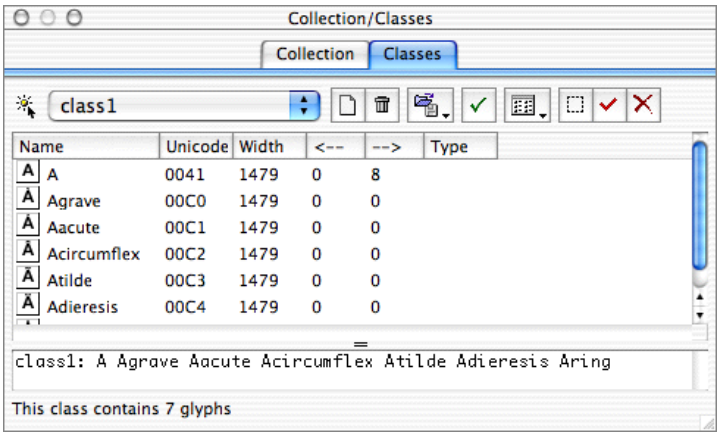
class1:

You **can manually enter glyph names** that you want to add to the class after a colon glyph:

class1: A Acaron

or you **can drag-drop glyphs** from the Font Window to append their names to the class.

After you add a few glyphs, the class panel will look something like this:



It's time to **rename the class**. Use the class definition panel and change "class1" to "_kernA". Click the Accept Changes button or the **Enter** key to accept the changes.

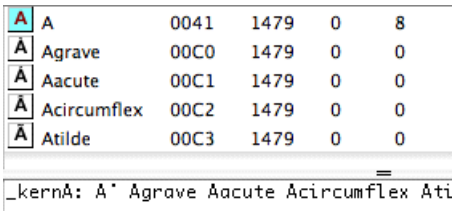
Note that the class name is changed in the class list:

Repeat the procedure to define a class for "T"-like glyphs.

Key Glyph

Kerning classes must include a declaration of the key glyph that is used as a kerning master for other glyphs in the class.

To define a master glyph, add a single quote after its name in the class definition text:

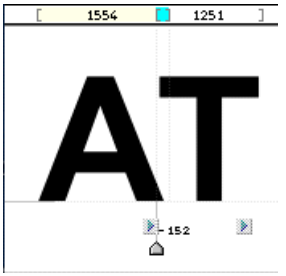

A 0041 1479 0 8
Agrave 00C0 1479 0 0
Aacute 00C1 1479 0 0
Acircumflex 00C2 1479 0 0
Atilde 00C3 1479 0 0
_kernA: A' Agrave Aacute Acircumflex Ati

After you accept the changes made in the class definition string, the key glyph will get a blue background in the preview panel.

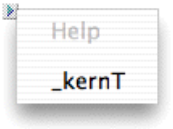
Editing Class-Based Kerning

To define kerning for a class you need to define the kerning for the key glyph of the class using the standard tools of the Metrics panel in kerning mode.

When you enter glyphs that are used as key glyphs into the kerning-editing string and select a pair of them for kerning editing you will see two buttons appear below the glyph images:



If you click on one of these buttons you will see a popup menu that lists all classes where the glyph works as a key glyph:



Select the class name and it opens for preview in the Classes panel.

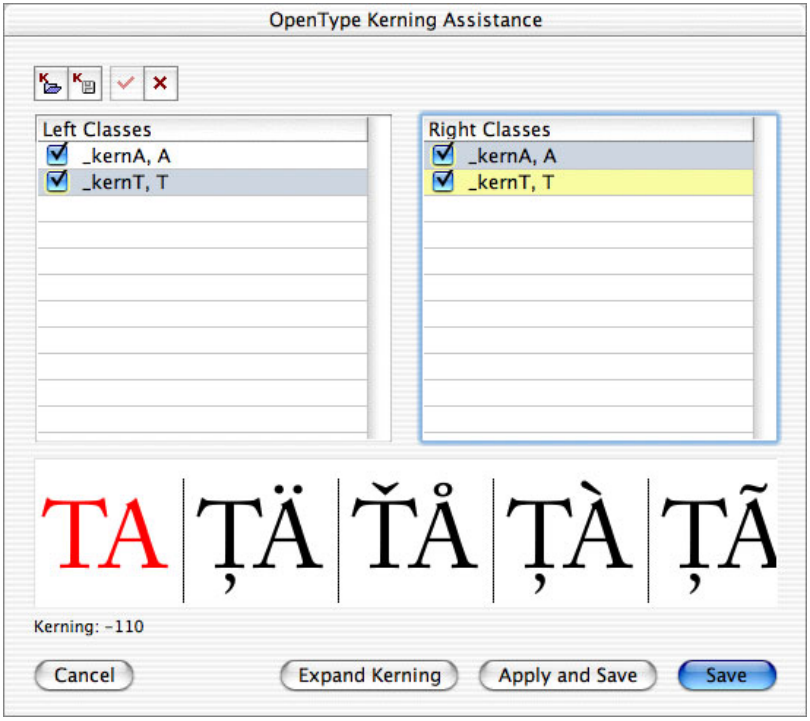
Similar buttons appear when you edit kerning using the Kerning Editor dialog box:

Their purpose is the same – to preview classes in which the currently selected glyph serves as a key glyph.

Kerning Assistance

FontLab has a special dialog box that can simplify the creation of class-based kerning. Before you start working with it, define some kerning classes (which have their name preceded by an underscore and one of the glyphs in the class is selected as a key glyph).

To open the Kerning Assistance dialog box use the **Tools/Kerning Assistance** command or click the button:



The dialog box has a toolbar at the top, two lists of kerning classes (for left and right glyphs in the pair) and a sample panel that shows the result of kerning of the pair.

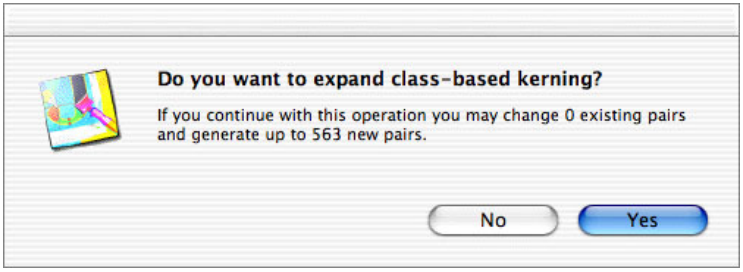
The idea of this dialog box is simple: use the check boxes in the left list to select kerning classes that contain glyphs that may be on the left in the kerning pair. Then check the kerning classes that contain glyphs that will be on the right in the pair in the right list. For example, if you have a kerning class that contains glyphs similar to 'K', it is unlikely that this glyph will be on the right in a pair, but it definitely will be used as a left glyph in pairs like "Ko".

The actual kerning positioning and the exact kerning value come from the kerning of the key glyphs of each of the kerning classes. This means that if you have an "_classA" and a "_classT" which have 'A' and 'T' as key glyphs in each class, respectively, you must define the kerning for the "AT" pair using the Metrics window. If you then checked "_classA" in the left list and "_classT" in the right list all glyphs in these classes would be kerned by the same value as the "AT" pair.

A **preview panel** in the bottom of the dialog box can preview the result of the class kerning. Drag to scroll the preview panel and see more sample pairs.

When you select a class in the left or in the right list, you may see some classes highlighted with the yellow background in the other list. This means that key glyphs of these classes are actually kerned with the key glyph of the class you have selected.

Sometimes you may need to convert class-based kerning into a plain kerning table. To do so, check the boxes of the class names that you want to "decompose" and click the **Expand Kerning** button. You will see the alert dialog box that tells how many new kerning pairs will be created:


Do you want to expand class-based kerning?
If you continue with this operation you may change 0 existing pairs and generate up to 563 new pairs.
No
Yes

Click **Yes** to proceed or **No** to not convert.

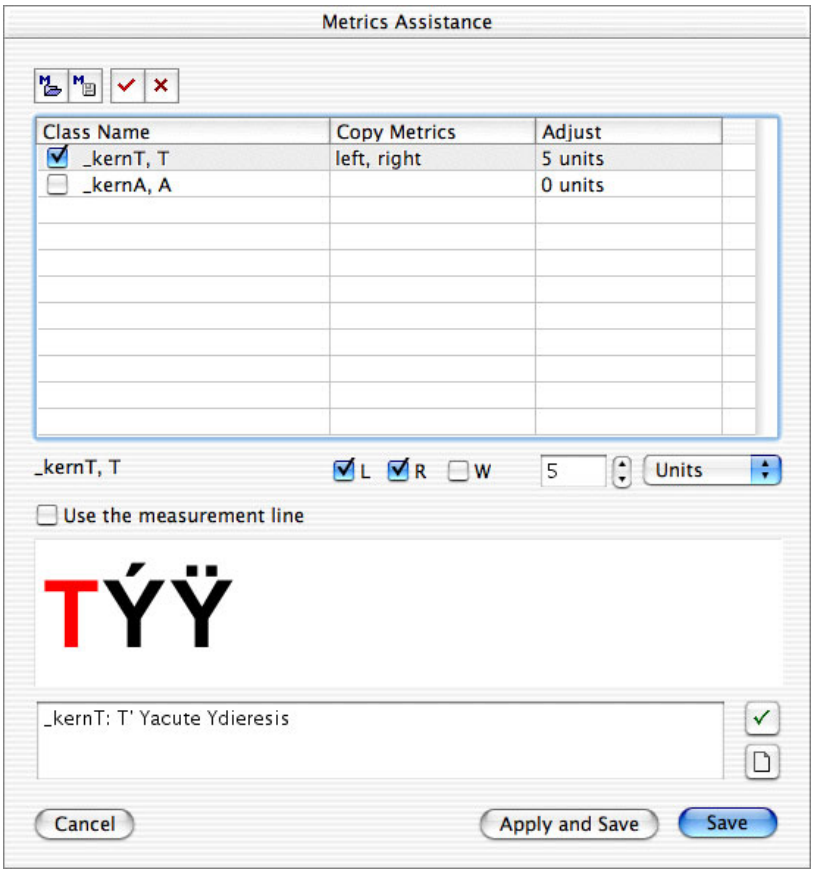
When you press the **Save** button the information you changed in the Kerning Assistance dialog box will be stored in the font. If you press the **Apply and Save** button, FontLab will generate the kern feature and class-based kerning will be applied if you build the font in OpenType (or TrueType) format. Please note that this Kerning assistance will not change any kerning in any of the pairs, it works exclusively with the class-based OpenType-style kerning. If you want to convert class based kerning to an old-style, plain kerning table, use the method described in the "OpenType Fonts" chapter.

Buttons on the **toolbar** for working with the kerning data:

	Open data file	Opens the data file with the kerning assistance information. FontLab can open data files saved by itself and kerning assistance files saved from Fontographer®. In latter case FontLab will automatically generate kerning classes and check them in the list similarly to Fontographer® Kerning Assistance dialog box.
	Save data file	Saves current kerning assistance information to a data file so you can apply it to a different font.
	Select all	Checks all items in the current list (left or right).
	Deselect all	Unchecks all items in the current list.

Metrics Assistance

Another way to use a kerning class is to adjust the metrics (advance width and left and right sidebearings) in the glyphs that belong to the class. Use the **Tools/Metrics assistance** command to open the Metrics assistance dialog box:



It has a toolbar, a list of the kerning classes, a command area, a preview panel and a class information panel that lets you change class definitions and define new classes.

Before you start working with the class metrics, define some classes. You can do that inside the Metrics assistance dialog box, but adding classes with the Class panel is much easier.

The typical procedure to define class-based metrics:

1. Select the class in the list.
2. Check the checkbox to the left of the class name to indicate that that class must be processed.
3. In the control area below the list check the metrics you want to copy from the key glyph of the class to the rest of the glyphs. It could be **L** for the left sidebearing, **R** for the right sidebearing and **W** for the advance width.
4. If you want, you can use the **Adjust** field at the right of the control area to change the metrics when they are copied from the key glyph of the class. Enter a positive or negative value and select the units – font units or percent of the original value.

Optionally you can measure sidebearings along the measurement line, which may be very useful if you are working with a serif font.

Use the preview panel to see the effect of your actions. To apply the new metrics click the **Apply and Save** button. Click on the **Save** button to store the metrics assistance information without actually changing the glyph metrics.

The editing field below the preview panel lets you change the class definition. Click on the button to accept changes. To create a new class, press the button.

Use the toolbar buttons to perform additional operations in the Metrics Assistance dialog box:

	Open data file	Opens the data file with the metrics assistance information. FontLab can open data files saved by itself and metrics assistance files saved from Fontographer®. In the latter case FontLab will automatically generate classes and import all information that is compatible with the FontLab metrics assistance feature.
	Save data file	Saves current metrics assistance information to the data file so you can apply it to different fonts.
	Select all	Checks all classes in the list.
	Deselect all	Unchecks all classes in the list.

Class-Based Kerning and OpenType Fonts

By default, FontLab will keep a class-based kerning structure when you open an OpenType font that has it. For each kerning class FontLab will automatically generate a class name and mark the first glyph in the class as the key glyph. This feature allows you to work with the huge kerning tables of some OpenType fonts in a comfortable environment that excludes most repetitive pairs.

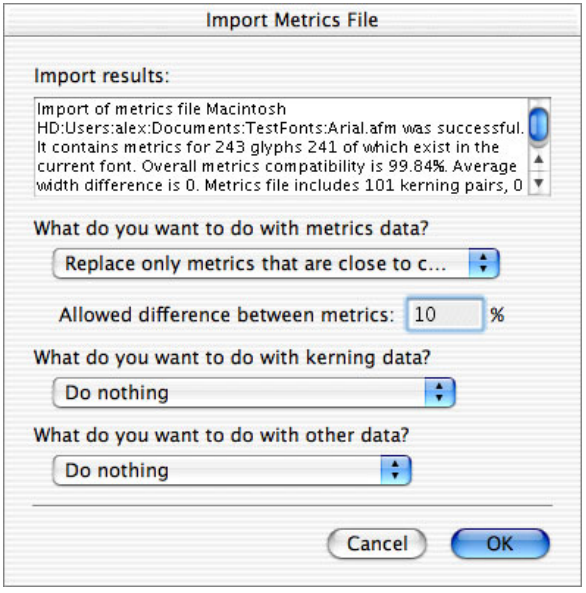
When you finish edit kerning pair values in the Metrics window you need to create a “kern” feature that will build kerning into the final OpenType font. You can use the special macro program available in the **Macros** submenu of the Metrics window context menu or you can simply remove the kern feature and FontLab will automatically generate a new one on export of the OpenType font.

Opening Metrics Files

FontLab allows you to import metrics and/or kerning information into the current font. Using this feature, you can create metric and kerning information once and use it in several similar fonts.

To import a metrics file into FontLab click the button on the Metrics window toolbar. You will see the standard Mac OS Open File dialog box. Select the metrics file that you want to import (in PFM or AFM format) and press the **Open** button.

The **Import Metrics File** dialog box appears:


Import Metrics File
Import results:
Import of metrics file Macintosh HD:Users:alex:Documents:TestFonts:Arial.afm was successful. It contains metrics for 243 glyphs 241 of which exist in the current font. Overall metrics compatibility is 99.84%. Average width difference is 0. Metrics file includes 101 kerning pairs, 0
What do you want to do with metrics data?
Replace only metrics that are close to c...
Allowed difference between metrics: 10 %
What do you want to do with kerning data?
Do nothing
What do you want to do with other data?
Do nothing
Cancel
OK

The topmost control contains a legend describing the metrics file that you are importing and its compatibility with the current font.

The options in the **Parameters** area let you select various metrics importing options:

What do you want to do with the metrics data:

Do nothing	Do not import metrics data from this file.
Replace all metrics in the current font	Import all metrics data (glyphs' widths and sidebearings) and replace the metrics data in the current font. We recommend that you use this option only if your font is very similar to the metrics file that you are importing.
Replace all metrics that are close to current	Replace only those metrics records that are similar to the imported metrics. The **Possible difference between metrics** option controls the allowed difference.
Replace metrics that are thinner than in the current font	These options are obvious.
Replace metrics that are wider than in the current font	

What do you want to do with the kerning data:

Do nothing	Do not import kerning data from the metrics file.
Completely replace kerning data in the current font	Remove all existing kerning pairs and replace them with pairs imported from the metrics file.
Add imported kerning data to the current font	Leave the existing kerning pairs unchanged but add new kerning pairs from the metrics file.
Add new kerning pairs but autokern them	Import information about the glyphs that form each kerning pair in the metrics file and apply an autokerning algorithm to these pairs.

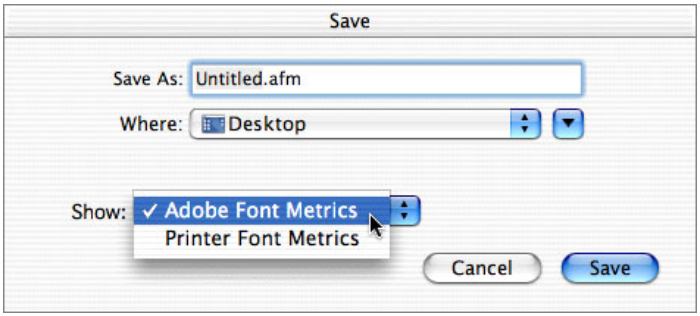
The **What do you want to do with other data?** option controls the font header importing option. FontLab can import the Font Info data from the metrics file and replace the current font info data if the **Replace this data in the current font** option is selected.

Note that when you open a metrics file while editing the metrics of a Multiple Master font only the metrics and kerning of the currently selected master will be replaced. Be careful.

Saving Metrics Files

When you export a font file in PC Type 1 font format the metrics files (in AFM and PFM formats) are automatically written. The TrueType font format includes all metrics information so it is not necessary to export additional files.

However, if you want **to export a metrics file alone**, you can always do so by using the Metrics Window. Just press the button on the Metrics Window toolbar. The Standard Save File dialog box appears:



Select the destination format (Adobe Font Metrics or Printer Font Metrics for single master fonts and Adobe MM Font Metrics or MM Font Metrics for Multiple Master fonts), and the destination folder. Enter the file name and press the **Save** button to save the metrics file.

Printing a Sample String

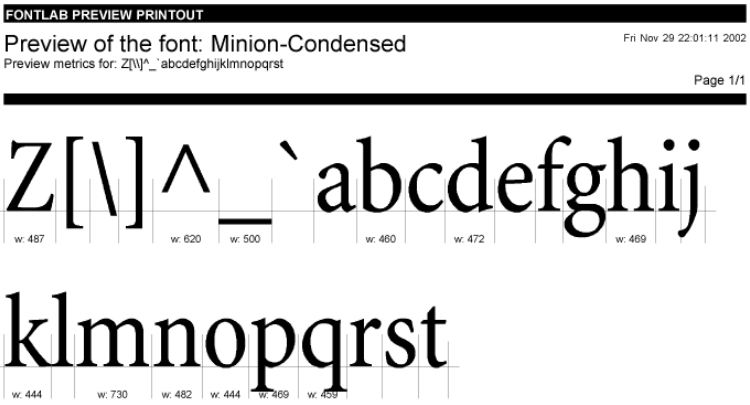
While you are in the Metrics Window you can print sample strings with or without metrics and kerning information. To do so select the **Print** command in the **File** menu.

You will see the standard Print dialog box where you select the printer and modify printing options. Press **Print** to begin printing.

FontLab will print the string that is currently selected for editing or previewing in the Metrics Window. The size of the printed string will be the same that is selected for previewing on the screen. The string may occupy more than one line or even more than one page on the paper if necessary.

If metrics or kerning editing mode is selected additional information will be printed below the baseline: in metrics mode – the widths of the glyphs and the glyphs' sidebearings; and in the kerning mode – the pair kerning values.

In the metrics editing mode kerning will not be applied to the printed string.



Transformations

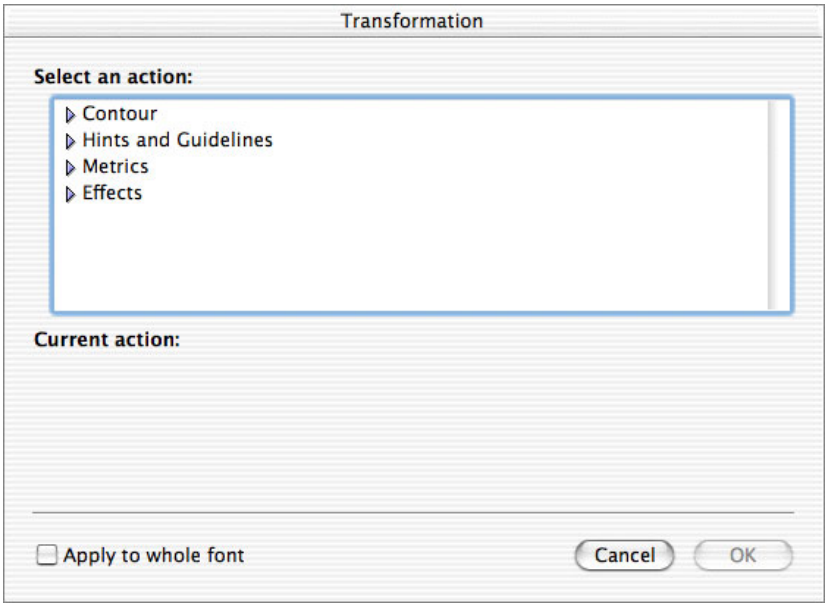
In FontLab you can transform glyphs in many ways. You can edit glyphs and glyphs' metrics manually using the Glyph and Metrics windows described in previous chapters. Or you can use FontLab's transformation filters to edit glyphs or metrics automatically. Transformations may be applied to one glyph, to a range of glyphs selected in the Font window, to a special set of glyphs (only to letters or only to digits, for example) or to a whole font. All the filters are carefully designed and always produce high-quality results that usually do not require manual control or correction.

In this chapter we will show you how to use the transformation filters and give a detailed description of each available filter.

6

The Transform Dialog Box

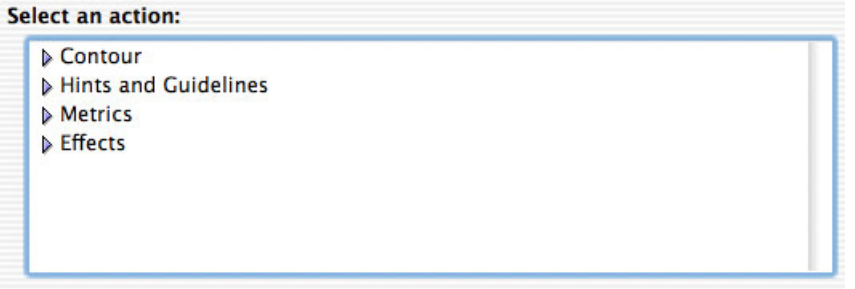
The easiest way to apply transformations is to use the Transform dialog box. It is accessible from the **Tools** menu while the Font or Glyph window is active. Select the **Transform** command from the **Tools** menu and you will see a dialog box:



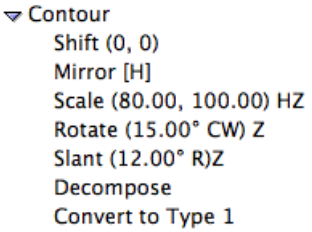
By default the box is empty and the dummy transformation action **Do not transform a glyph** is selected in the actions list.

If you open this dialog while the Glyph window is active the transformation filter will be applied only to the glyph currently open and is undoable. If you open it while the Font window is active then the transformation will be applied to all selected glyphs.

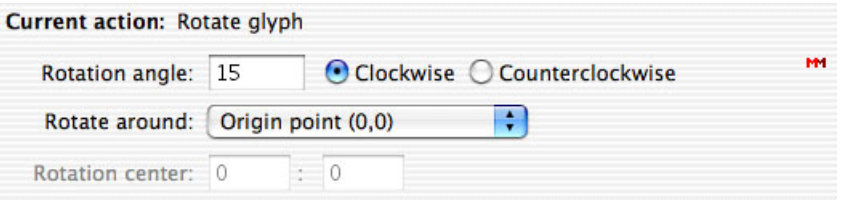
To select a filter use the *Select an action* list:



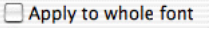
Expand one of the categories to see all the filters:



Select an action and you will see an action customization panel appear below the list:



The contents of the customization panel depend on the action selected. Note the red MM mark in the right top corner of some customization panels. This mark means that the currently selected action may be applied to a single master of a Multiple Master font. All actions that do not have this mark are not compatible with MM fonts. You can use them, but all the masters will stick together and you will lose the “multiple-masterness” of your font.

Set the 

option to apply the transformation to all the glyphs regardless of your selection in the Font window.

After you select a transformation action and set its options, press the **OK** button to begin transformation. If you are transforming more than one glyph a warning message will appear telling you how many glyphs you will modify and asking you to confirm the transformation. Transformation of many glyphs is not undoable, so it's a good idea to save your font before taking this action.

Below you will find a detailed description of each available transformation action. A description of the more sophisticated **Transform Range** dialog box, where you can build a transformation program that can include many actions, finishes up this chapter.

Transformation Actions

There are four groups of actions:

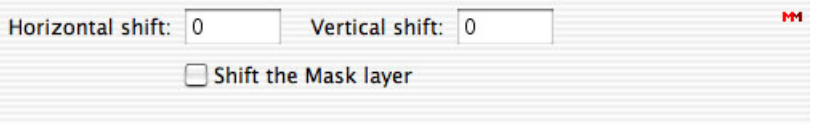
Outline	The outline of a glyph is transformed.
Hints and Guidelines	Actions that are concerned with hints and links.
Metrics	Metrics information is transformed (includes automatic metrics generation).
Effects	A set of effects that can be applied to glyphs.

Outline Transformation

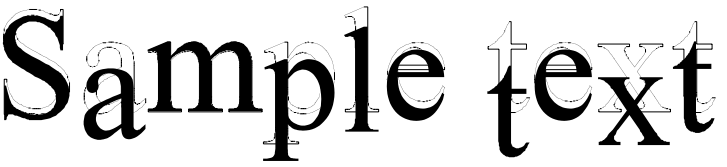
Here is a list of all the outline transformation actions:

Shift	Shifts the glyph's outline
Mirror	Mirrors the glyph vertically or horizontally
Scale	Scales the glyph proportionally or non-proportionally
Rotate	Rotates the glyph
Slant	Slants the glyph
Decompose	Decomposes a composite glyph
Convert to Type 1	Converts an outline to the Type 1 (3rd-order curves) format
Convert to TrueType	Converts an outline to the TrueType (2nd-order curves) format
Reverse all	Reverses the outline direction of the glyph
Connections	Automatically detects the connection types between a glyph's graphical primitives
Extremes	Automatically inserts points at extreme points on curves
Remove Overlap	Removes overlapping parts of the glyphs' outline
Make Master	Automatically calculates the 4th master when 3 other masters are known
Optimize	Optimizes the glyph outline with a custom set of options
Blend	Blends the outline and mask layers.

Shift



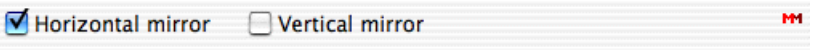
This action shifts the outline of the glyph in the vertical and/or horizontal direction. Here is a sample of a font with some glyphs shifted in the vertical direction:



You can also shift glyphs in the vertical direction in the metrics mode of the Metrics window: hold the **Shift** key and drag the glyph.

Use the **Shift the Mask Layer** control to shift the mask layer together with the outline or to leave it untouched.

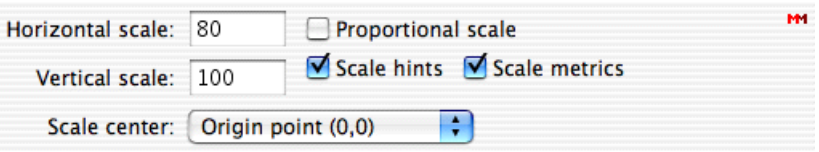
Mirror



Here is the result of this simple transformation:

The letters in the word “Sample” were mirrored horizontally and the letters in the word “text” – vertically.

Scale



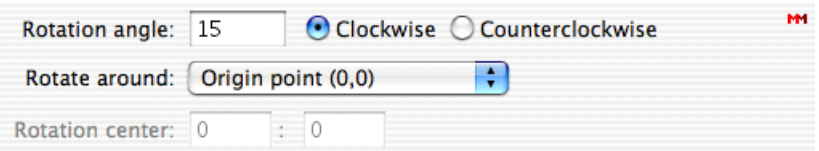
This action lets you scale your glyphs proportionally or non-proportionally. Enter your desired vertical and horizontal scale factors in the edit fields. Switch on the **Proportional scale** option to keep the vertical and horizontal scale factors the same.

Switch off the **Scale hints** option to avoid scaling hints along with the glyph's outline. Scaling hints' width is not always precise so if you scale hints with the outline you sometimes find that some hints now miss the nodes that they were supposed to hint. We recommend converting hints to links before this transformation to keep the proper width and position of hints.

Here is an example of this transformation (the letters of the word "Sample" were scaled 80% horizontally and the letters of word "text" were proportionally scaled to 120% of original size):

Sample text

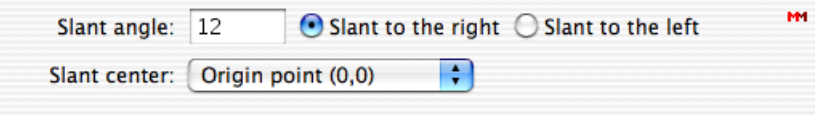
Rotate



This transformation action simply rotates glyphs. You can set the rotation angle, the position of the center of rotation and the direction of rotation.

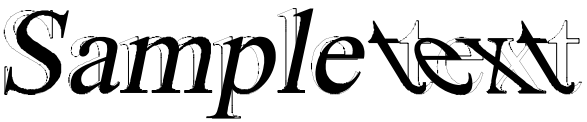
You can rotate glyphs around the zero point, around the center of the glyph's bounding box or you can specify a point that will be used as the center of rotation. Here is an example of the same rotation transformation around different center points:

Slant



This action slants glyphs. It's the quickest way to make an oblique version of your font. Just apply this transformation to all the font's glyphs and correct the Font Info settings to let the operating system know that this font is now oblique.

Here is a sample of Slant transformation ("Sample" is slanted 12 degrees to the right and "text" is slanted 30 degrees left):



Decompose

This action is the equivalent of the **Decompose** command from the **Glyph** menu. It adds the scaled and shifted outlines of component glyphs to the composite glyph where they are used. Applied to a whole font it makes it free of composites.

Convert to Type 1

This action converts all TrueType (2nd-order) curves used in the selected glyphs into Type 1 (3rd-order) curves. Use this command if you want to manually prepare a TrueType font for conversion into Type 1 format. Do not forget to check the direction of the contours to make the glyphs compatible with Type 1 format requirements.

Convert to TrueType

This action is the reverse of the previous one. It converts Type 1 curves into TrueType curves. Usually it is used to prepare a Type 1 font for manual TrueType hinting.

Reverse all

This action reverses the direction of all the contours of the glyph's outline. It can also automatically detect the direction of contours and correct them if necessary. Just select the option that you want.

Connections

Use this action to reset and recalculate the types of connections between outline primitives. It is useful after major modification of an outline when you want to review how outline primitives are connected.

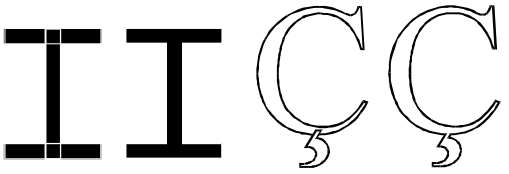
Extremes

This command is the equivalent of the **Nodes at extremes** outline action that was described in the "Glyph Window" chapter. Use it to automatically insert nodes at curves' extreme points as is required by the Type 1 specification.

Remove Overlap

This action removes overlapping parts of the glyph's outline. It also sets the direction of all the contours to counterclockwise (as it required by the Type 1 specification). Use this transformation as the final step to prepare glyph outlines for hinting.

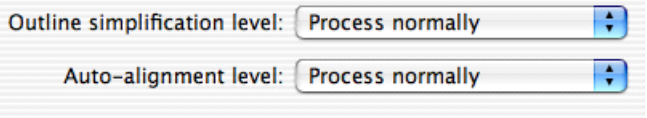
Several examples of this transformation:



Make Master

This transformation takes three known masters of a four-master Multiple Master font and calculates the fourth master. It uses a simple algorithm of linear extrapolation, so the results are not usually precise. But they may give important information about the proportions of the fourth master, so you can get a close approximation of what the fourth master should look like.

Optimize



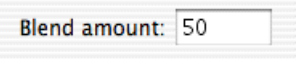
With the Optimize action FontLab tries to automatically adjust the outline to remove unnecessary elements and correct others.

With the action options you can control the optimization process:

Outline Simplification level	Controls the curve removal feature, from “do not simplify outline” to “extreme”. The bigger value you choose the more curves FontLab will try to remove/smooth.
Auto-alignment level	Controls the auto-alignment feature in a range from “do not align” to “extreme”.

Auto-alignment automatically corrects relative position of lines and curves, for example, if two adjacent curves are almost smoothly connected, but not precisely, auto-alignment will correct that.

Blend



This transformation blends the mask and outline layers. A single option lets you choose the “position” of the intermediate design between the mask and outline layers:

Outline and mask layers

On the example above outline layer is black weight and mask layer is regular. Below are samples of the Blend transformation with different values of the parameter:

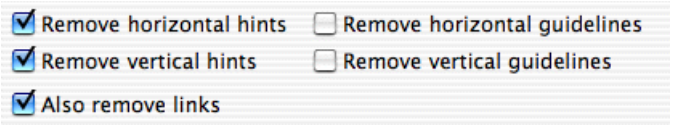
10% *50%* *80%*

Hints and Guidelines Transformation

Hints transformation actions let you automate some hinting actions:

Remove hints/guides	Lets you remove hints and links or guidelines
Autohint	Automatically generates hints
Convert to instructions	Converts Type 1 hints to editable visual TrueType instructions
Autoreplace	Automatically generates a hint replacement program
Convert to links	Converts all hints to links
Convert to hints	Converts all links to hints
Drop TT Hints	Removes all TrueType hints, visual or imported.

Remove hints/guides



Use this action to remove hints and links or guidelines in selected glyphs. This command is the equivalent of the **Remove Hints** and **Remove Guides** actions that were described in the “Glyph Window” chapter.

Autohint

Analyses a glyph’s outline and generates hints for the glyph. This action uses autohinting options that can be set in the Type 1 page of the Preferences dialog box. Please refer to the page 334 for a description of the autohinting options.

Convert to instructions

Use this action to convert Type 1 hints to the editable TrueType visual instructions. Do not forget to prepare the outlines of the glyphs for TrueType hinting using the Convert to TrueType transformation action. You can find a detailed description of this process in the “Hinting” chapter.

Autoreplace

This action automatically builds a hint replacement program for the overlapping Type 1 hints. Refer to the chapter “Hinting” for more information about hinting and hint replacement.

Convert to links

Use this action to convert Type 1 hints to links. FontLab will analyse the outline of the glyph and try to make links that will replace the hints. Refer to the “Glyph Window” chapter for information about hints and links.

Convert to hints

This action is the reverse of the previous one. It converts all links to hints. Because this operation is always possible (conversion of hints to links is not), you can be sure that it will replace all links by hints.

Drop TT Hints

This action removes all TrueType hints, including visual (manually created) and imported, from the source TrueType font.

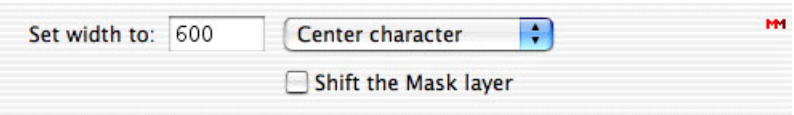
Metrics Transformation

These transformations let you automatically set metrics, calculate metrics in glyphs, increase or decrease a glyph's sidebearings and width, and center glyphs in their width vector.

Available metrics transformations are:

Set Width	Sets a fixed width and aligns the glyph within the width vector
Set Sidebearings	Sets or changes sidebearings' values
Center glyph	Centers a glyph in the width vector
Autospacing	Automatically calculates the glyph's metrics using the same algorithms that are used in the Metrics window
Adjust metrics	Changes sidebearings and kerning by the given value in font units or in percentage.

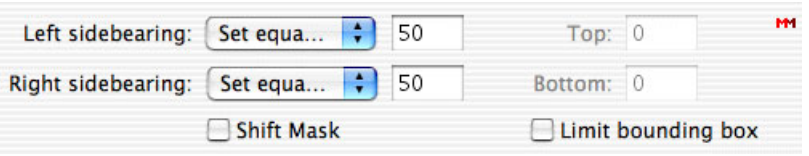
Set Width



This action lets you set a fixed width for all glyphs that are selected in the Font window. It is the fastest way to make a monospaced font: select all glyphs, open the Transform dialog, set the desired width and press the **OK** button.

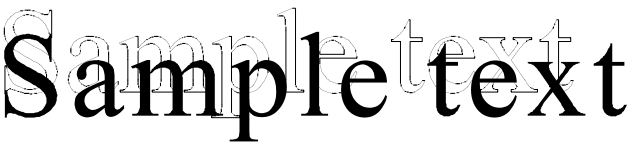
In the options list you can choose what to do with glyphs that are thinner than the requested width vector. A glyph may be aligned to the left or right margins or it may be centered in the width vector.

Set Sidebearings



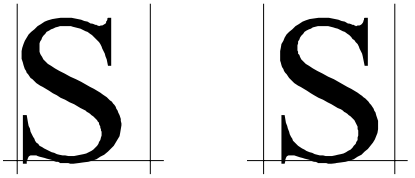
Use this action to change the sidebearings' values of the glyphs. You can set new values for the left or right sidebearings or change these values by entering the amount in font units. So if you think that your font needs some more white space, just select this action, choose the **Increase by** option in the popup menus and enter the value by which you want to increase the sidebearings.

Here is a sample of increased glyph widths:



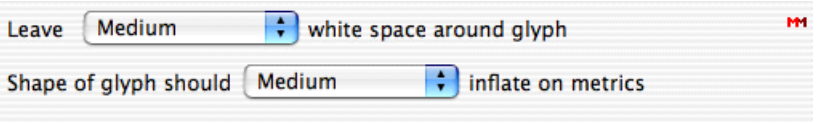
Center glyph

This action simply centers the glyph in the width vector:



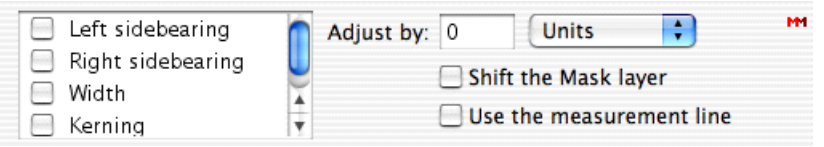
Before centering *After centering*

Autospacing



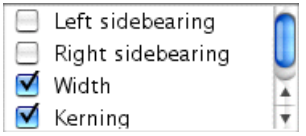
This action analyses the glyph's outline and automatically calculates its sidebearings. It uses the same algorithm that is used to automatically calculate metrics in the Metrics window. Refer to the "Editing Font Metrics" chapter for more information about the autospacing algorithm and options.

Adjust Metrics



This action lets you change metrics or kerning by some given value or percentage of the original value.

In the options area select the metrics that you want to change:



Enter the amount by which you want to adjust the values and select the units (which can be font units or a percentage of the source value). Enter a positive number to increase metrics distances or kerning or a negative value to decrease them.

Normally metrics are measured from the bounding box. You can change this, however, if you want, to measure from a measurement line by using the check box.

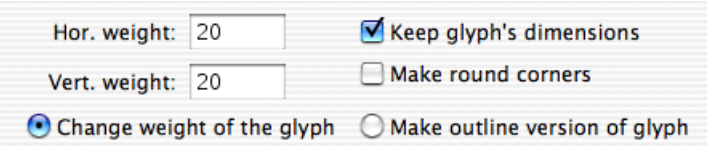
A measurement line is horizontal line (defined in the Glyph Window) that defines a "slice" which is used to measure the distance from outline to sidebearings. Please, refer to page 307 in the "Glyph Window" chapter.

Effects

Increase your font library with this set of professionally designed effects. From outline to 3D-shadow, to gradient fill – these transformation filters always produce good results.

Bold/Outline	Increases the glyph's weight or creates an outline version of the glyph
College	Makes a double outline version of the glyph
Shadow	Generates a drop shadow
3D Extrusion	Makes a "3D" version of the glyph
3D Rotate	Makes an illusion of a glyph rotated in 3D space
Gradient	Generates an illusion of a gradient fill
Random	Randomly moves nodes
Envelope	Applies one of the predefined envelope transformations.

Bold/Outline



This is one of the most used actions in FontLab. With it you can precisely change the weight of the glyph's stems, make an outline version of the glyph or change the contrast:

Sample

Original glyphs

Sample

Bold 20 units in both directions Keep glyph's dimensions is ON

Bold 20 units in both directions Keep glyph's dimensions is OFF
Note that the size of all the glyphs has changed

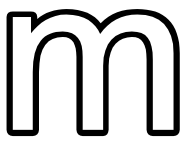
Bold 20 units in the horizontal direction

Outline 20 units in both directions. Make round corners is OFF

Enter the horizontal and vertical values that will be used to increase (positive numbers) or decrease (negative numbers) the weight of the outline.

Switch on the **Keep glyph's dimensions** check box to scale the glyph so the weight-increasing effect will be compensated.

Switch on **Make round corners** to make rounded corners in the new outline:



Note that the weight-changing values are in font units, so the visual effects of this effect depend on the font's UPM value.

College

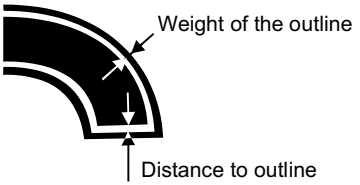
Distance to outline: 10

Weight of outline: 10

The best way to explain this effect is to see a sample of it:

Sample

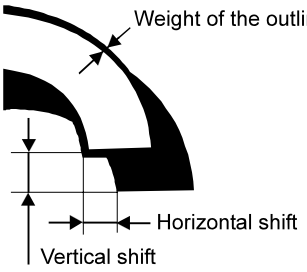
The parameters of this effect are very simple:



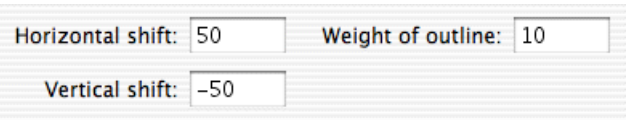
Shadow

This is a very nice effect that can save you a lot of time. Those who have tried to make a shadow font manually know what a miracle this transformation performs:

The parameters set the shift of the shadow (positive values are to the right and up) and width of the outline:



3D Extrusion



This action is similar to the Shadow action but it simulates a 3D thickness of the glyphs:

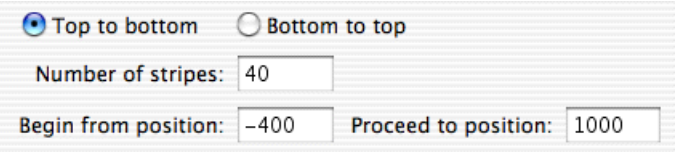
The parameters of this action are the same as in the previous section.

3D Rotate

With this action you can “rotate” your glyphs in “3D” space:

The parameters of this action set angles of rotation for glyphs around imaginary axes. **Z Rotation** means rotation around the vertical axis. **Y Rotation is** around the horizontal axis. The vertical axis goes through the middle of the glyph.

Gradient



Here’s what you can do with this effect:

As parameters of this effect you can set the number of stripes that appear on your glyphs, the starting and finishing line of the effect and the direction of the gradient.

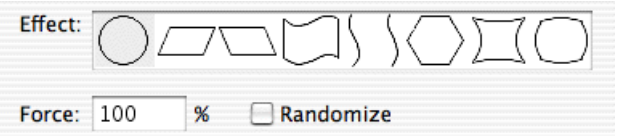
The starting and finishing lines can be manually set to let you customize this effect and make it look the same in all glyphs. Note that these values are set in font units so they are relative to the font’s UPM value.

Random

This effect randomly shifts a glyph's nodes. It is especially interesting when combined with other effects, like Gradient:

You can control how much the nodes are shifted. You can set the same value for both directions if the **Proportional offset** option is on or you can customize the values separately.

Envelope



This last effect lets you apply one of many predefined transformations to several glyphs at once. Refer to the "Glyph Window" chapter for more information about the Envelope effect. Here is an example of what you can do with this effect:

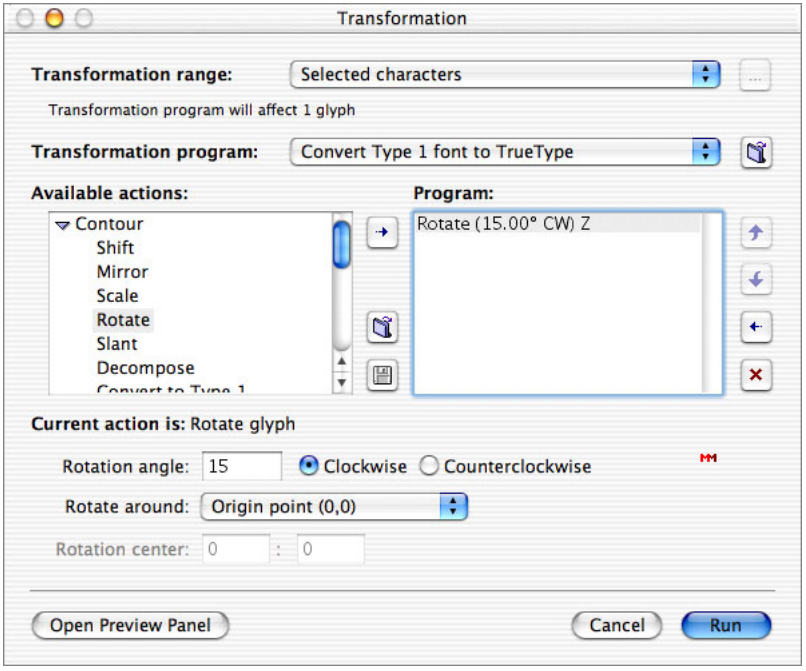
Arial font after "circle" envelope with -30% Force value

The **Randomize** option applies some random changes to the effect to make an even more interesting result.

Transform Range Dialog Box

There is a more advanced method of applying transformation effects to glyphs. With it you can apply many transformation filters at once, transform a subset of a font's glyphs, see an instant preview of the programmed transformation and even save transformation programs for future use.

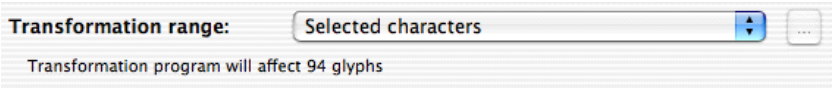
To open the Transform Range dialog box select the **Transform Range** command in the **Tools** menu. Note that this command is available while any of the windows is open. You will see a big and complex dialog box:



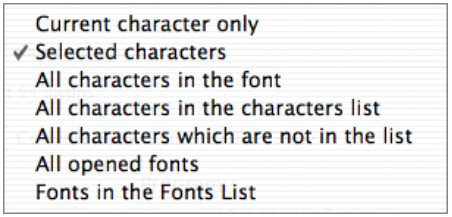
This dialog box has several areas that control different options of the transformation.

Transformation Range

In the top part of the dialog box a **Transformation Range** area is situated:



Controls in this area let you select which glyphs you want to transform. Open the popup menu and you will see the available options:


Current character only
✓ Selected characters
All characters in the font
All characters in the characters list
All characters which are not in the list
All opened fonts
Fonts in the Fonts List

Current character only	Transformations will be applied only to one current glyph: the "blue" glyph in the Font window or the current glyphs in the Metrics or Glyph windows.
Selected characters	Transformations will be applied to all the glyphs that are selected in the Font window.
All characters in the font	The whole font will be transformed.
All characters in the characters list	Only glyphs that are enumerated in the glyphs' list will be transformed (see below).
All characters that are not in the list	Only those glyphs that are not in the list will be transformed. Thus if the list includes all the digits and you select this option, all glyphs except the digits would be transformed.
All opened fonts	Transformation program will be applied to all glyphs in all opened fonts.
Fonts in the Fonts List	Transformation program will be applied to all glyphs in all fonts that are added to the fonts list (see below).

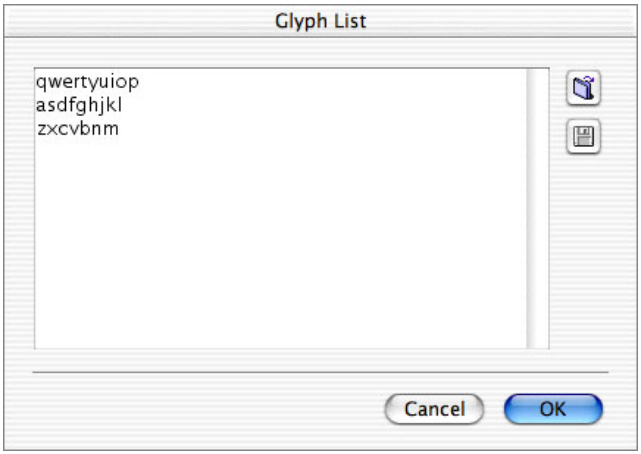
If you are editing a Multiple Master font the master selection popup menu will appear below the range list and you can select the master that will be transformed. Note that only transformation actions that have the red MM mark in the right-top corner may transform a selected master. All other actions always transform all masters.

Glyph List

You may select which glyphs will be transformed by entering a list of glyphs. This is handy for repetitive or recurrent transformations.

To create a glyph list select the **All characters in the characters list** or **All characters that are not in the list** options in the ranges selection popup and press the ... button (which will be enabled).

You will see a dialog box:



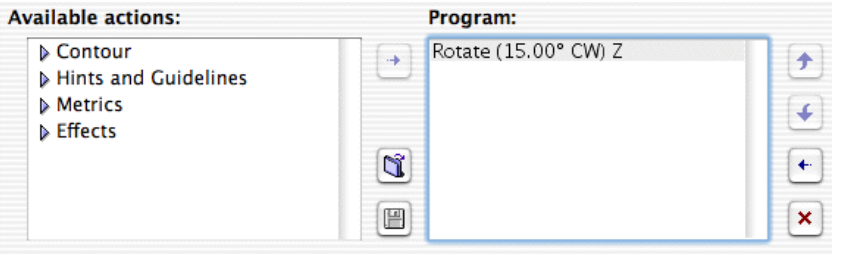
Enter all the glyphs that you want to transform into the **Glyph list** editing field. You can use special character commands to enter characters that are not included in the standard Roman character set used in Mac OS by default. Use a '/' prefix to enter a glyph's name or use a decimal code and a "/#" prefix to enter a glyph's Unicode index.

Use the **Open** button to open any text file and use it as a glyph list and the **Save** button to save the current glyph list in a text file.

Note that if you open the Transform Range dialog box while the Metrics window is active the glyph list will be copied from the Metrics window's sample string and the **All characters in the characters list** option will be automatically selected, so you can instantly apply transformations to the set of glyphs that are previewed in the Metrics window.

Transformation Program

In the middle of the dialog box there are two list boxes:



The left list box, called **Available actions** includes the names of all available transformation actions. The actions are grouped in categories for easier selection. The right list box previews the current transformation program.

You may add as many actions as you want to the transformations program; delete actions from the program; or rearrange actions to make them execute in proper sequence.

To add an action to the transformations program select the action (use the ▷ and ▽ icons in the **Actions list** to expand and collapse the action categories) and press the → button. Or double-click an action name.

The action will be added to the bottom of the program with its default parameters.

To adjust an action's parameters select the action in the program list and enter new parameters in the **Action's options** area. The actions' parameters dialog boxes are described in the previous sections.

To remove an action from the program select the action that you want to remove and press the ← button. Or double-click the name of the action in the program list.

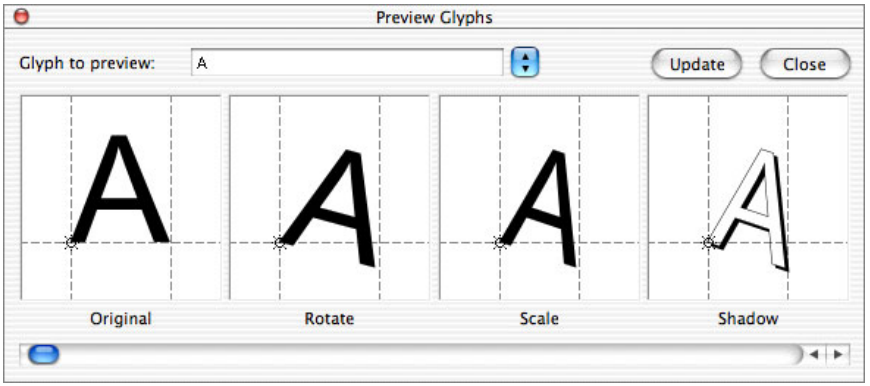
To remove all actions from the program press the × button.

To move an action one step up in the program select it and press the ↑ button or press the ↓ button to move an action one step down.

Using the Preview Window

When you make a program that includes several actions you can see an instant preview of the transformation program that you have made.

Press the **Open Preview Panel** button to open a preview window:



You see that this panel includes a sample glyph that is incrementally transformed. You can see the results of each step of the transformation so it's very easy to control how your transformation program works.

With the **Glyph to preview** list box you can select the glyph that is used to preview the transformations.

Use the scroll bar at the bottom of the window to see further samples of the transformation actions program.

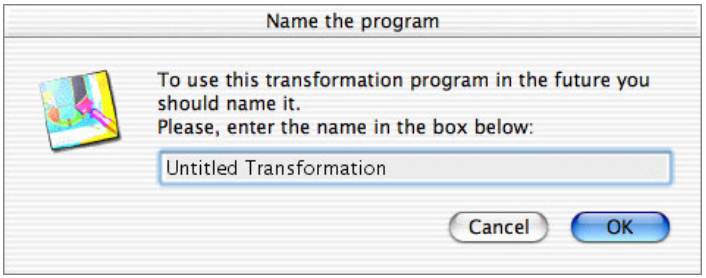
Press the **Close** button to close this window.

Saving and Opening a Transformation Program

You can save the transformation program that you created and you can open and use previously saved transformation programs.

To save the transformation program press the button. You will see a standard dialog box where you can select a folder and enter a file name for the program. Note that if you save a program in the Programs directory it will automatically appear in the **Transformation program** popup menu the next time you open the Transform Range dialog box.

After you press **Save** in the Save File dialog box a new dialog box will ask you to enter a name for the transformation program:


Name the program
To use this transformation program in the future you should name it.
Please, enter the name in the box below:
Untitled Transformation
Cancel
OK

This name will be used to identify the program and it will appear in the **Transformation program** popup menu.

To open a previously saved transformation program press the button and select one of the programs in the standard Open File dialog box.

A faster way to open transformation programs is to use the **Transformation program** popup menu. The names of all transformation programs saved in the FontLab/Programs folder will appear in this list:

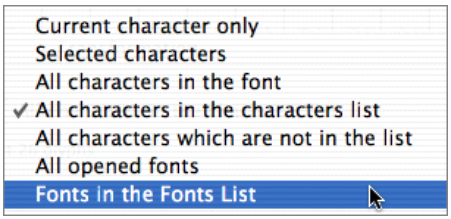
Select the program that you want to open and press the button. You will see the program appear in the Program list box.

Transforming Fonts

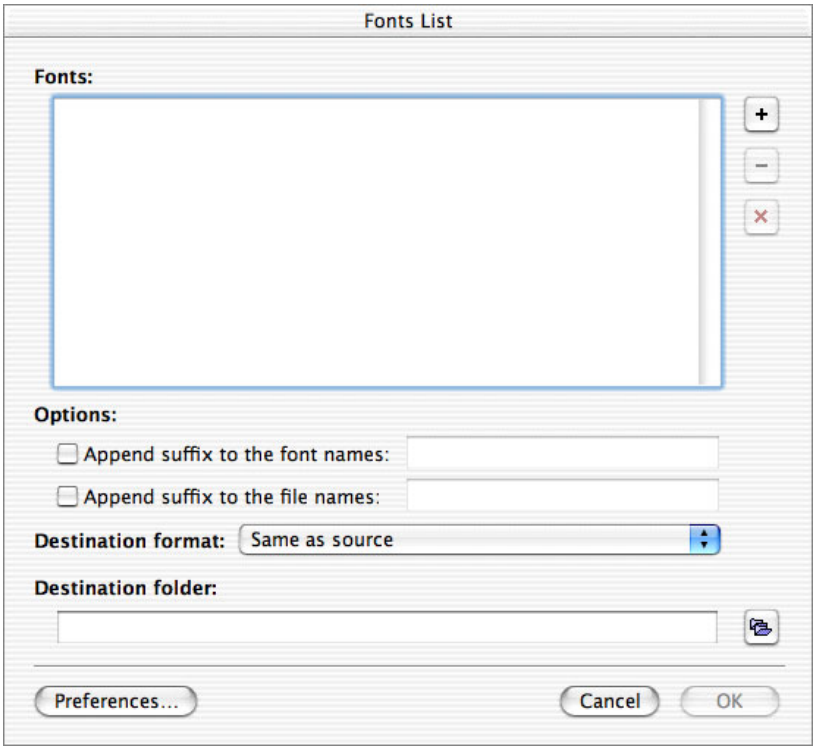
With the Transform Range dialog box you can apply a transformation program to multiple fonts at once. The easiest way is to apply it to all fonts that are open in FontLab. If that is not enough, you can run the transformation program in "batch mode", processing multiple fonts that are not open in FontLab.

To select fonts for transformation:

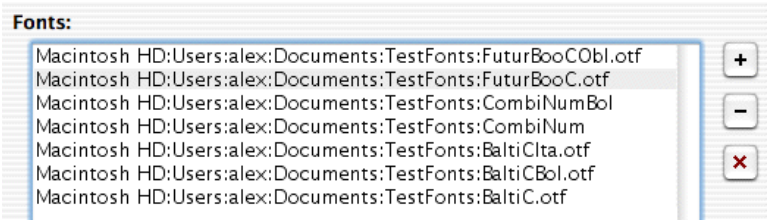
1. Select "Fonts in the Fonts List" as the Transformation range:


Current character only
Selected characters
All characters in the font
✓ All characters in the characters list
All characters which are not in the list
All opened fonts
Fonts in the Fonts List

2. Click on the ... button to the right of the Transformation range control to open the Fonts List dialog box:

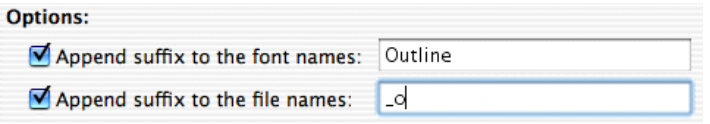


3. Click on the + button to add fonts to the list using the standard File Open dialog box. Note that you may select multiple font files to add:

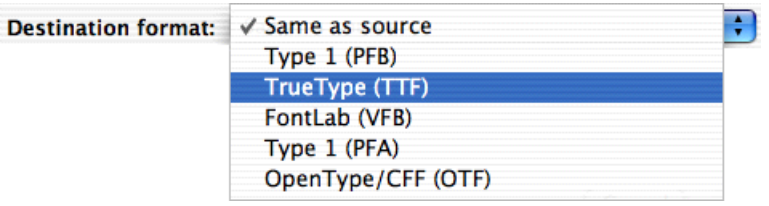


4. To remove a font from the list select it and click on the − button. Click the × button to remove all fonts from the list.

5. Use the Options controls to define the optional suffixes that will be added to the font name and to the file name during transformation:



In the same area you can select the destination format in which the font will be exported:

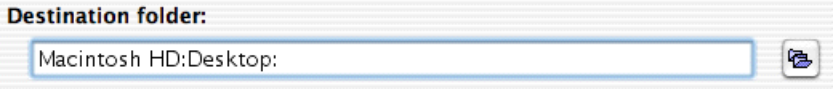


Use the "**Same as source**" choice to leave the font's format unchanged.

The font's customized export options (if source fonts are in FontLab format) or the currently selected export options will be used to build the destination font.

Note that you are not required to have any transformation program and you can use this batch transformation feature to convert fonts from one format to another.

6. Enter the path to the destination folder or use the button to select it from the one of the existing folders:



7. Click **OK** to complete the definition of the list.

After you start the transformation FontLab will open fonts from the list; apply the transformation program; adjust the font and file names if specified; and save the resulting fonts in the specified formats to the selected folder on your disk.

Hinting

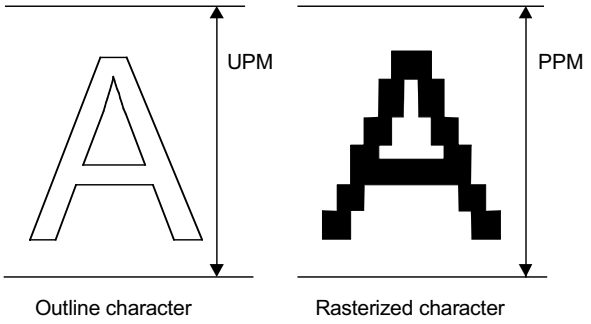
This chapter is about hinting Type 1 and TrueType fonts. Hinting is a rather “technical” stage of the modern font design process. It is the stage where art truly meets technology. You need a lot of technical information to make well-hinted fonts and this chapter will give it to you.

7

Font Scaling, PPM

One of the most important features of outline fonts is that they can be used on many different output devices – from computer monitors to image-setters. Because character outline shapes are defined as sequences of lines and curves it is easy to scale outlines to any size and resolution. However, almost all output devices have discrete elements arranged in a regular rectangular raster (grid) and the images that these devices produce are constructed using these discrete cells. Each cell in an output image has integer coordinates and is called a "pixel" (picture cell). On a computer monitor these are individual fluorescent dots. On a printer they are dots of toner or ink.

To measure scaled outlines in a resolution-independent way, it is convenient to define a quantity called *Pixels Per eM* (PPM). This is the number of pixels that can fit into the font's height. From the "Glyph Window" chapter you know that font height is a basic font measurement unit equal to the *Units Per eM* (UPM). In TrueType fonts UPM is usually equal to 2048 and in Type 1 fonts to 1000.

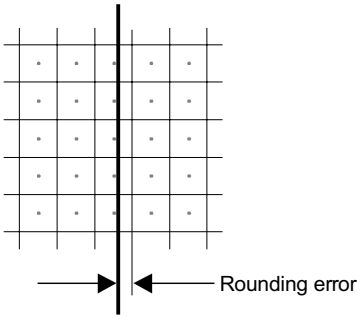


So, to scale a font to render at a specific point size on a device with a specific resolution we take the resolution and point size to calculate the PPM value. Then we scale all the outline characters by multiplying by the PPM/UPM coefficient.

Coordinate Rounding, Gridfitting

Output devices take the vector outline of a glyph and “rasterize” it. That is, they calculate from the outline data where they need to place each pixel to get an accurate output representation of the glyph. Since the final output is on a discrete raster (i.e. a grid of numbered pixels), the scaled pixel coordinates need to be rounded somehow to integer values.

For example, if you have an outline point with coordinates (120, 100) and scale it down 7 times, you will get the coordinates (17.1429, 14.2857). After rounding to the closest integer values, the resulting coordinates will be (17, 14) and so the rounding error will be 0.1429 (0.84%) pixels for the horizontal coordinate and 0.2857 (2%) for the vertical coordinate. If we instead scale this point down 13 times, then the scaling errors will be 2.5% for the vertical coordinate and 3.8% for the horizontal coordinate. You can see that the rounding error increases as the size of the final outline is reduced.

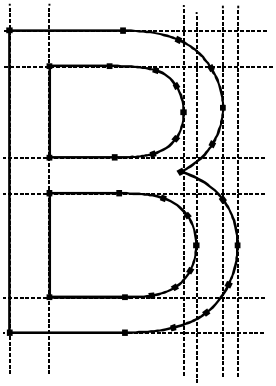


To minimize rounding errors font rasterizers use special algorithms that slightly change the scaled outlines to get better results on devices with low and medium resolution. This process is called *gridfitting*. Algorithms that gridfit outlines use additional information stored with an outline’s definition. These instructions are referred to as *hints*. Hints usually define the most important proportions of characters, the positions of critical elements of characters, and a set of rules for outline modification.

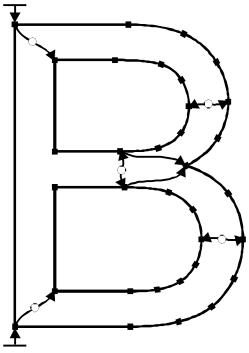
For perfect-looking fonts it’s not enough to define the characters' outlines, you must also provide hints. The process of specifying the hints is quaintly called *hinting*.

TrueType And Type 1 Hints

The two most commonly used font formats are Type 1 and TrueType. However they use very different hinting instructions and it is not always possible to automatically convert Type 1 hints to TrueType hints.



Type 1 hinting



TrueType hinting

In Type 1 fonts, hints define the most important dimensions in the characters, like the position and width of the crossbar of the letter 'H.'

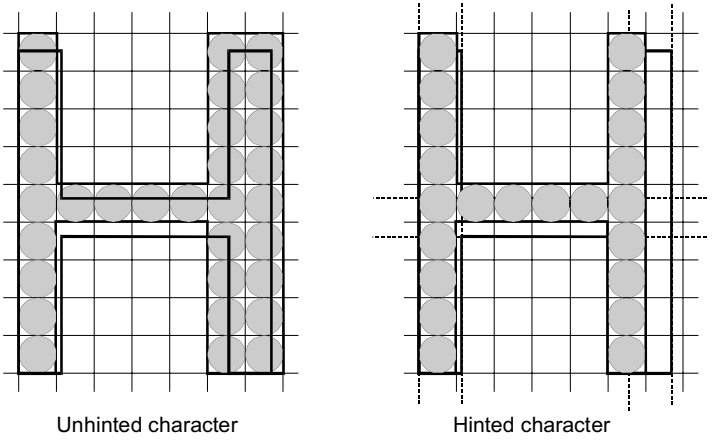
Font hints in TrueType (usually called instructions) directly control the movement of points and the rounding of point coordinates. A TrueType hinting program is written in a special programming language. This makes TrueType hinting very flexible and powerful but also too complex to program directly. Usually a smaller set of higher-level instructions is used to define hints. These instructions are compiled to native TrueType hinting language during font export.

Type 1 Hints

As you know from the "Glyph Window" chapter, there are two kinds of Type 1 Hints – *font level hints* and *character-level hints*.

Font-level hints define important vertical positions in the font, the most commonly used stem widths and some other important data that helps control the hinting process.

Character-level hints are used to declare the position and width of the most important character elements. The most common use for hints is to declare the position and width of character stems. These hints are scaled with the outline in the rendering stage, but due to their independence from the outline, they help to maintain the same stem widths for all stems of a certain width, independent of how it happens to fall on the discrete raster:



Unhinted character Hinted character

Notice that in the unhinted character the outline falls on the grid such that two rows of pixels would be turned on for the right vertical stem. The hint forces the right stem to become the same width as the left stem in the hinted character.

In the following sections we will discuss font-level hints and the process called *hint programming* that is required at times.

Font-Level Type 1 Hints

Font-level hints are used to keep important character elements similar at all PPM sizes.

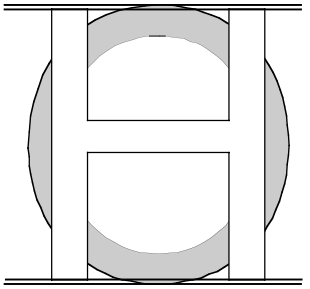
There are three types of font-level hints:

Alignment Zones	Positions and width of important heights
Standard Stem Widths	Widths of the most commonly used stems
Control Data	Controls the hinting process.

All font-level hinting is set in the **Alignment Zones** page of the **Font Info** dialog in FontLab.

Alignment Zones

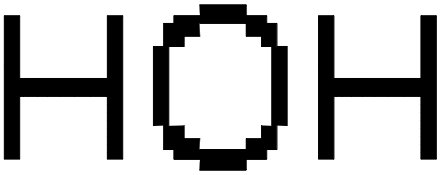
Alignment zones are typically used to perform a process known as overshoot suppression. Rounded characters and characters with sharp ends usually are created a little bit larger than “flat characters”:



Notice that the top and bottom of the O extend just a little beyond the top and bottom of the H. This is called overshoot. It is necessary to compensate for a visual effect that makes rounded characters look slightly smaller. Usually the overshoot height is set to 3-4% of character height. However, at small PPM size, this value may be rounded to one pixel.

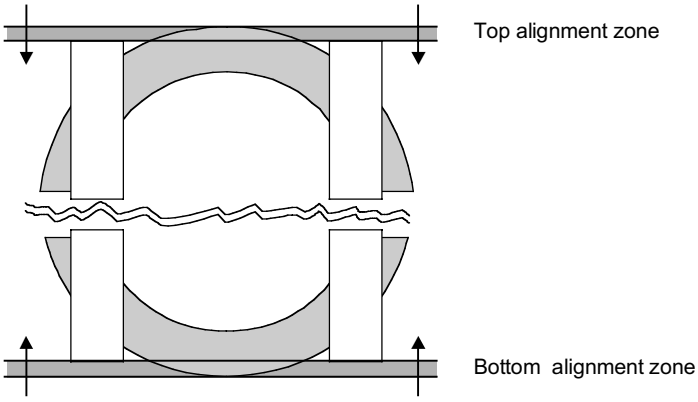
When the PPM is small, one pixel may be 15% of the character height or even more. Here’s how it happens:

Assume that the topmost position of the H character is 700 units and the top position of the O is 715. At 12 PPM (1000 font units scaled to 12 pixels), the rounded height of the H will be 8 pixels. The height of the O will be 9 pixels. One pixel difference at this height means 8%. Much more than the original 2%!



To avoid such an excessive difference between the rasterized heights of the two types of characters, overshoots are suppressed and the size of O is forced equal to the height of H at small PPM.

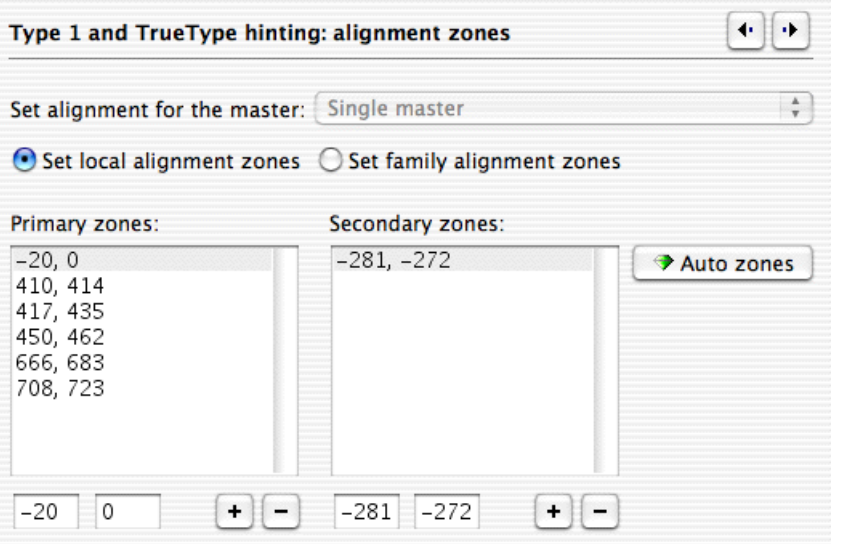
This is done by declaring alignment zones that define the bottom and top positions of the zone (in our example the height of H and O) and the alignment direction (bottom or top):



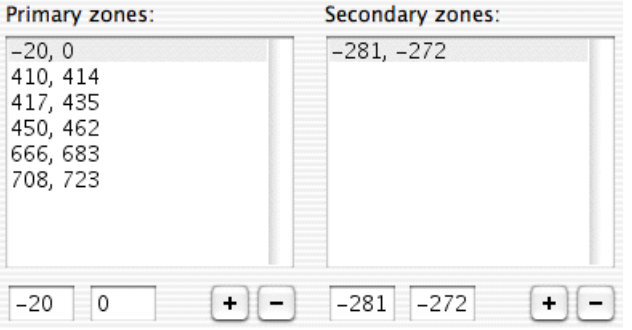
At small PPMs all points that have vertical positions inside the zone will be aligned to the primary line (i.e. moved in the direction of the alignment zone).

Editing Alignment Zones

To set or edit alignment zones open the **Font Info** dialog box and select the Hinting settings item in the list at the left:



There are two list boxes where alignment zones may be set: the **Primary zones** list and the **Secondary zones** list:



In Type 1 terminology primary zones are called *BlueValues* and secondary zones *OtherBlues*.

BlueValues include one bottom alignment zone, the so-called *baseline zone,* and up to 6 top alignment zones. The baseline zone is used to control bottom overshoots that have to be aligned to the baseline.

OtherBlues includes up to 5 bottom alignment zones.

To add a new alignment zone, press the + button below the list.

To edit the position of the zone, select the zone you want to edit in the list and edit it in the edit fields below the list.

To remove an alignment zone, select the zone you want to remove from the list and press the − button below that list.

You can **see a preview of the zones** by switching on the Alignment Zones layer with the **View/Show Layer/Alignment zones** command.

When zones are visible on screen, you can preview changes you make to the zones by pressing the **Apply** button in the Font Info dialog box.

You can **edit zones in the Glyph window**, using the edit tool. Make sure they are not locked by the **View/Lock layers/Alignment zones** command.

Press the Auto zones button to automatically calculate alignment zones in the **Primary zones** list box.

How FontLab Calculates Alignment Zones

To calculate alignment zones in the BlueValues list, FontLab finds characters with overshoots and characters that are flat in the position of the overshoot. Then it measures the top and bottom vertical positions of these characters and detects a zone. Examples of such characters are: 'o' and 'x', 'O' and 'H', 'p' and 'g', and so on. FontLab tries to find many different characters from different languages, so it is usually able to locate some examples.

Alignment zones are also used in TrueType manual and automatic hinting.

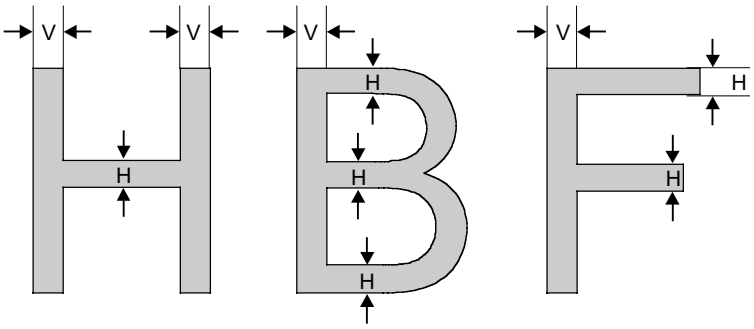
Family Alignment Zones

To support the common appearance of fonts that belong to the same font family the Type 1 hinting system allows so-called FamilyBlues, alignment zones that are used in the whole font family. Typically the alignment zones of the regular weight are used as Family Blues in all members of the family.

To set family alignment zones switch on the **Set family alignment zones** check box. Then edit the alignment zones as usual. To return to editing "local" alignment zones switch on the **Set local alignment zones** check box.

Standard Stem Widths

Typically many characters in a font use the same few standard stem widths. As examples, let's take the H, B, and F characters shown below. All of them have the same width for the straight vertical stems and the same width for the horizontal stems:



The most widely used stem widths are stored in the font header in order to force the rasterizer to render these stems at the same width.

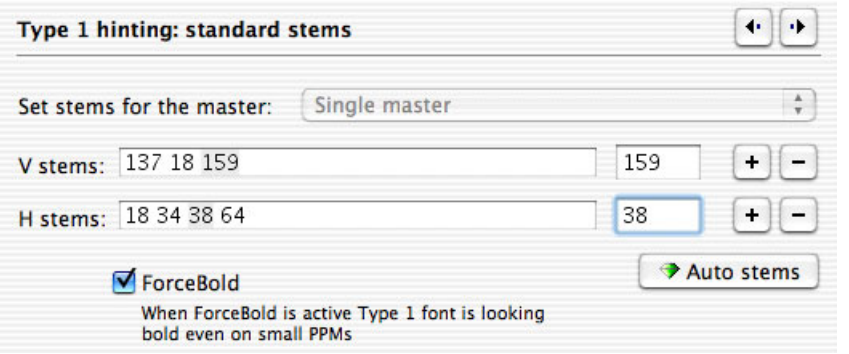
This information is used to control at what character size the rounded stem width goes from one to two pixels and from two to three pixels. A step from one to two pixels means a 100% width increase and a step from 2 to 3 pixels a 50% increase. This means that near this value rounding errors will be maximal and control over stem widths will be necessary.

If one stem has a width of 74 units and another a stem width of 76 units and the UPM is 1000 units, then at a PPM of 20 pixels the first stem will be rounded to 1 pixel and the second stem to 2 pixels. Scaled back to the original coordinates, this difference will be 50 units! That is clearly too much for an original difference of only 2 units.

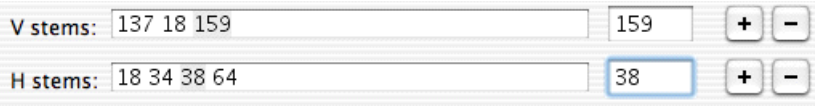
Standard widths work with stem hints. When the width of a hint is close to one of the standard widths, the rounded width of the hint (and the real stem outline) will be forced equal to the width of the rounded standard stem.

Editing Standard Stems

Standard stem widths are controlled through the Standard Stems page of the **Font Info** dialog box:



Stem width controls are in the center part of this page:



There are controls for vertical and horizontal standard stem widths. All available stems appear in the horizontal lists. You can select any stem just as you would in normal, vertical list controls.

To add a stem to the list of the standard stems press + at the right side of the stems' list.

To edit a standard stem width, select it using the mouse and edit its value in the edit field to the right of the list.

To remove a stem from the list, select it and press the − button.

Note that FontLab will sort stem widths in ascending order when you close the **Font Info** dialog box.

StdHW, StdVW, StemSnapH and StemSnapV Parameters

From the Type 1 font specification you may know that in Type 1 fonts two types of standard stem widths are used: Standard Width and Stem Snap Width. There is one standard width for each direction and up to 10 stem snap values. In FontLab these values are united in the stem list. StdHW and StdVW are taken from the first records in the stem lists. StemSnapH and StemSnapV records are the remaining records in the stems' list.

Standard stem widths are also used in TrueType hinting.

FontLab also has a faster way to append stems to the list of standard stem widths. Any vertical or horizontal hint may be used as a source of stem width information. Just point the Edit tool at the hint, **Ctrl**-click and select the **Define a Stem** command in the popup menu. If this command is not accessible, it means that this stem is already in the list.

To automatically calculate standard stem widths press the

button.

How FontLab Calculates Standard Stems

FontLab can calculate standard stem widths only if some characters in the font have Type 1 hints, so it converts links to hints first (in memory) using the most important glyphs in the font.

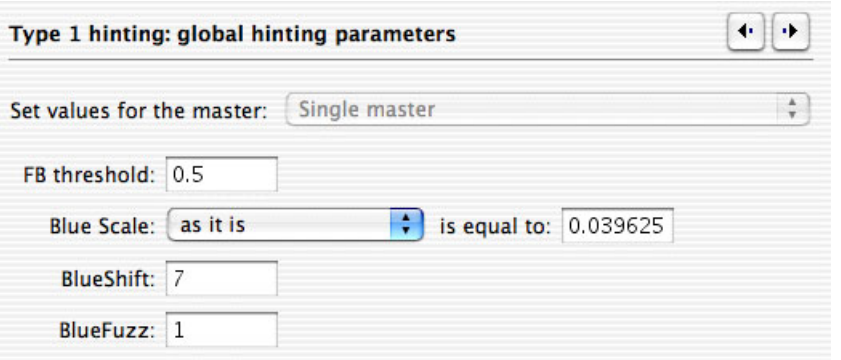
1. It builds a table of all hints that are used in the font, sorts this table by frequency of usage and selects the most frequently used hints.
2. All selected hints are then compared with these most frequently occurring stem widths and hints with widths that are close together are combined into a single record.
3. The list is then sorted again.
4. The most frequently used elements are then selected and used as standard stems.

Additional Control Data

Some additional data may be set to control the hinting process:

BlueScale	Controls PPM when overshoot suppression is switched off
BlueShift	Gives more precise control over overshoot description (see below)
BlueFuzz	Expands alignment zones in both directions.

You can set all these values on the **Additional Hinting Parameters** page of the Font Info dialog box:



BlueScale is the PPM size at which overshoot suppression is switched off. If PPM is less than BlueScale, then overshoot suppression is applied. If it is equal to or exceeds BlueScale, overshoot suppression works only if the distance from the aligned point to the base line of the alignment zone is less than the **BlueShift** value and the scaled distance is less than half of a pixel.

The BlueScale value is stored in Type 1 fonts in a very strange format, but in the **Alignment Zones** page you can set it using one of three different ways: directly, in the form that the Type 1 specification describes (i.e. it looks like a floating point number), as a PPM size, or as a point size on a device with 300 DPI resolution. Use the **BlueScale:** combo box to select the BlueScale editing method and edit it in the **is equal to:** edit field.

BlueScale Formulas

The "actual" value of the BlueScale value is calculated as:

$$BlueScale = \frac{\left(PPM - 1.63333\right)}{800.0}$$

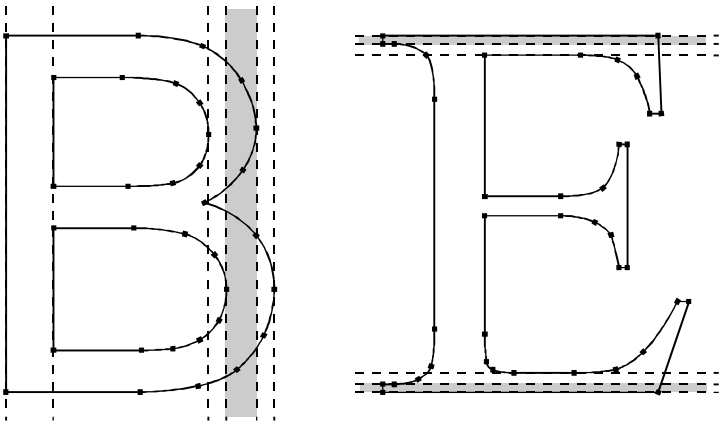
The **BlueFuzz** value allows you to expand the action range of the alignment zones in both directions. Thus if you have defined a zone like (700-715), and BlueFuzz is equal to 2, then the actual zone used will be (698-717). This is usually used when you are not sure that you correctly set all the alignment zones or when the characters are not all precisely aligned. The normal value of this parameter is 0 but by default it is set to 1. It is not recommended to use the BlueFuzz value other than 0.

TrueType hinting algorithms do not use BlueScale, BlueFuzz and BlueShift values.

Stem Hint Programming

Now that you know everything about font-level hinting and know how to set and edit stem hints (from the “Glyph Window” chapter). It’s time to talk about hint programming.

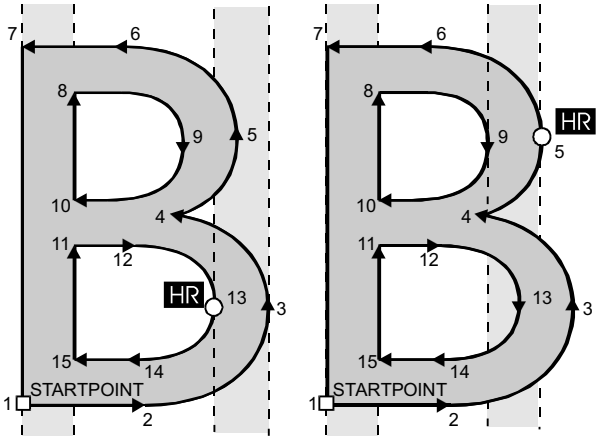
The Type 1 hinting system requires that stem hints not overlap each other. Because stem widths have no length limits and are applied to the whole character, sometimes it is necessary to give special instructions to hints in order to avoid such overlaps:



Overlapped hints

In the left picture we have two vertical overlapping hints and in the right picture two pairs of horizontal overlapping hints.

Each of these hints should, however, work only on part of the outline:



As you can see, we have two hint sets and three outline segments: from point 1 to point 4, from point 5 to point 12 and from point 13 to point 15.

So we must switch active hint sets as the rasterizer proceeds along the contour to match the part of the outline that the hint sets apply to. We need to have some program that will switch off the right hint in segment 2 (after the rasterizer has passed point 3, where the right hint applies and before it gets to point 5 where the left hint applies) and switch it back on in segment 3 (before it gets to point 13 where the right hint applies again). The same program should switch on the left hint in segment 2 and switch it off again in segment 3. This way only one of the two hints is active at any one time.

This program is called a *hint replacement program* and the process that it performs is called *hint replacement*.

FontLab will generate hint replacement programs for every character where necessary but you can also set hint replacement programs manually if necessary to get the best possible results.

You can see the state of the character's hints and hint replacement program instantly in the Font Window. There is a small mark in the left-bottom area of each character cell. If this mark is present, it means that a hint replacement program is present. If the mark is **green** it means that the program is OK, that is, there are no overlapping hints. If this mark is **red** it means that the character has overlapping hints.

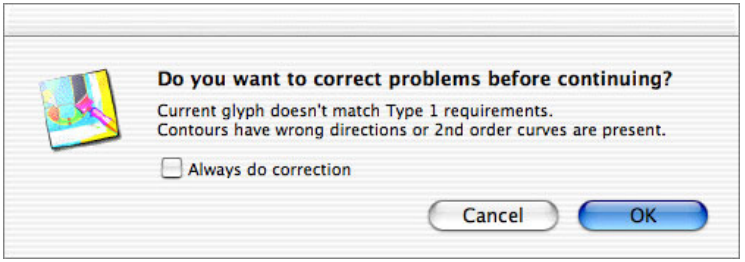
The hint replacement program will be removed if you insert or delete a hint or a node, or apply any transformation operation to the character. Consequently, we recommend that you set advanced hinting information only *after* you finish editing the character's outlines, at the last stage of font development.

Type 1 Hinting Tool

Use the Type 1 Hinting Tool to create a hint replacement program.

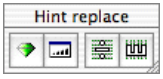
To activate the Type 1 Hinting tool select **Type 1 Hinting** in the **Tools/Operations** menu. Or just click on the button in the Tools or Operations toolbar.

If the glyph that you are hinting contains TrueType curves FontLab will open the **warning dialog box**:


Do you want to correct problems before continuing?
Current glyph doesn't match Type 1 requirements.
Contours have wrong directions or 2nd order curves are present.
Always do correction
Cancel
OK

Click **Cancel** to avoid activating the Type 1 hinting tool or **OK** to convert the glyph to Type 1 curves. Check the **Always do correction** option to convert all glyphs that are opened for Type 1 hinting.

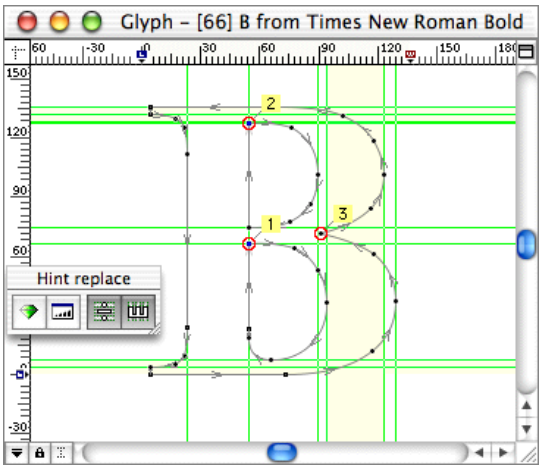
When the Type 1 hinting tool is successfully activated you will see a small Hint Replace toolbar:



The buttons on the toolbar mean:

	Auto	Automatic hinting
	Preview	Opens the preview panel
	3-Hint-H	Allows horizontal triple hints when pressed
	3-Hint-V	Allows vertical triple hints when pressed

When the Type 1 hinting tool is active, the contents of the Glyph window change:



Here is a brief description of what appears in the editing field:

Thick green lines	Currently active set of stem hints
Thin gray lines	All other hints
Yellow areas	Overlapping zones of hints
Black contour	Active part of contour
Green "HR" marks	Hint-substitution points
Yellow marks	Startpoints of contours with their number
Arrows	Direction of contours.

There are active and inactive contour segments and active and inactive stem hints. Active hints work when the active part of an outline is processed by the rasterizer. The hint replacement points separate contour segments from each other.

You can **select different active parts of the outline** by clicking on the outline segment that you want to make active using the mouse. The active segment always appears in black and the hints that apply to that segment appear in green.

Inserting And Removing Replacement Points

The contour segments to which you may assign hints lie between two hint replacement points (green marks). To define a new contour segment you add a new replacement point. Remember that the node where you put the hint replacement mark will be the first node to which the hint set is applied.

To add a hint replacement point:

1. Position the mouse cursor on the node where you want to set the hint replacement mark.
2. **Ctrl**-click the mouse button. You will see the popup menu with only one command: **Add replace point here**. Select it and a new replacement point will appear.

When you insert a new replacement point FontLab automatically selects the hints that should be set in the new contour segments.

To remove a hint replacement point:

1. Position the cursor on the node that has an "HR" mark.
2. **Ctrl**-click the mouse and select the **Remove this replace point** command from the popup menu.

FontLab will combine hints from the two segments, trying to make an optimal hint selection. It will not let hints overlap.

Adding and removing hints

FontLab automatically chooses the hints that should be included in the hint set that belongs to the contour segment beginning from the new hint replacement mark. To modify this hint set you can add or remove hints from it.

To add a hint to the hint set:

1. Position the mouse cursor on the hint that you want to add. The cursor should be between two hint lines.
2. Ctrl-click the mouse. The selected hint will be highlighted and you will see the popup menu with two commands: **Add this horizontal hint**, and **Add this vertical hint**. Only one of the two commands is available depending on which hint you want to add. If the popup menu does not appear, it usually means that you tried to add a hint that will overlap one of the hints that is in the current hint set.

   If the popup menu appears but includes a different set of commands, it means that you missed and selected the wrong hint, or no hint at all.

   "Wrong hints" are hints that overlap any of the hints that are already active in the current segment.

To remove a hint from the hint set:

1. Locate the mouse cursor somewhere between the lines of the hint that you want to remove. Currently active hints are shown in green.
2. Ctrl-click the mouse and select the **Remove this horizontal** (or **vertical**) hint command from the popup menu.

Editing Hints

With the Type 1 hinting tool you can also add, edit and delete Type 1 hints and links.

To add a new hint, hold the **Cmd** key and drag it from the vertical or horizontal ruler line in the left and top parts of the Glyph Window and position it where you want the hint to be.

To add a new link, select the **Add new vertical/horizontal link** command from the **Tools/Actions/Hints and guides** menu or from the popup menu that appears if you **Ctrl**-click the mouse on an empty area of the Glyph Window. Click the mouse cursor on the first node of the pair you want to link and drag it to the second node and then release it.

To edit a hint or link, position the mouse cursor on the hint's or link's line, press the mouse button and move it to the new position. Refer to the description of the Edit tool on page 329 for more information about hint editing.

To remove a hint or link, position the mouse cursor on one of the hint's or link's lines and **Ctrl**-click the mouse. Select the **Delete** command in the popup menu.

Select the **Reverse** command from the same menu to reverse the hint's direction. Correct direction of hints is left-to-right and bottom-to-top. Incorrectly directed Type 1 hints may cause the problems with some older font rasterizers, so FontLab will correct the hint direction automatically during the Type 1 font generation.

Select the **Properties** command in popup menu to open the hint (or link) properties panel.

There is a faster way **to open the properties panel**: hold the **Cmd** key and click on one of the hint lines with the mouse button.

Note that if you delete a hint you will remove it completely, not just the reference to it as in the case of the **Remove** command described in the previous section.

Autoreplacing

The easiest way to build a hint substitution program is to click the button on the Hint Replace toolbar. We recommend that you begin from the automatically generated hint substitution program and make adjustments as necessary.

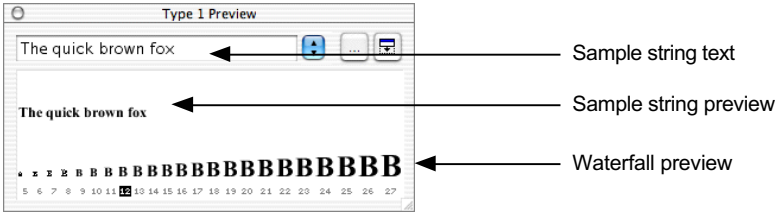
Preview Panel

The Type 1 Hinting Tool has a preview panel that shows how Type 1 hints will affect the character's appearance. The preview panel uses the Adobe Type Manager rasterizer to preview the font under Mac OS 9, so you may be assured that in the final font you will get exactly the same look. Of course you need to have some version of Adobe Type Manager (ATM) installed in order to use the Preview panel.

Under Mac OS X the preview panel uses ATSUI (Apple Type Services for Unicode Imaging).

To show the preview panel press the button in the command panel.

The preview panel consists of two fields: a standard edit field where you can enter a sample string, and a preview window, where two types of preview appear:



In the waterfall preview you can see the current character in various PPM sizes. Note, that if grayscale rendering (text smoothing) is enabled, you will see grayscale previews at some PPMs.

In the top part of the preview window a sample of any character can appear. We recommend using this sample area to compare the rasterizing results of different characters in a font. You can change the PPM at which the sample string is previewed very easily: position the cursor on the PPM preview in the waterfall range and double-click the mouse. You can also browse the PPM values by pressing the **Cmd-PgUp** and **Cmd-PgDn** shortcuts on the keyboard.

To see a preview of special characters that are not accessible directly from the keyboard use the usual FontLab rules for entering special characters (i.e. enter the character name after a slash '/' character – use two slashes to enter a slash: "//").

FontLab updates the preview panel every time you change something in the hint replacement program. This may be a slow process on some computers, but you can hide the panel; edit the hint replacement program; and then switch the panel back on to see the results of your work.

How the Type 1 Preview Panel Works

For Type 1 preview FontLab uses either Adobe Type Manager (ATM) in Classic environment or ATSUI under Mac OS X. When FontLab starts in Classic, it makes a connection with ATM and prepares it for work. When the Type 1 preview panel is open and something is changed in the character FontLab performs the following operations:

1. FontLab creates a very small Type 1 font that includes the empty (".notdef") character, the current character and all characters from the sample string.
2. All characters in this font are mapped to codes starting at 0x3F (63) – the code of the question character. The empty character is not mapped.
3. This font is exported in Type 1 font format and a suitcase file is created for it. The font name is set for this font in such a way as to avoid conflict with names of any of the installed fonts.
4. The font and suitcase files are saved in temporary directory.
5. The font is installed in System (using ATM commands under Classic).
6. All previews are made with this font. Previews in the Preview panel can use grayscale output if it is available in the system.
7. When the font is no longer needed it is uninstalled and the font and suitcase files are removed.

Expanding the Preview Panel

The Preview panel may work in “closed” or “expanded” mode. To open the panel click on the . You will see the preview panel expand:

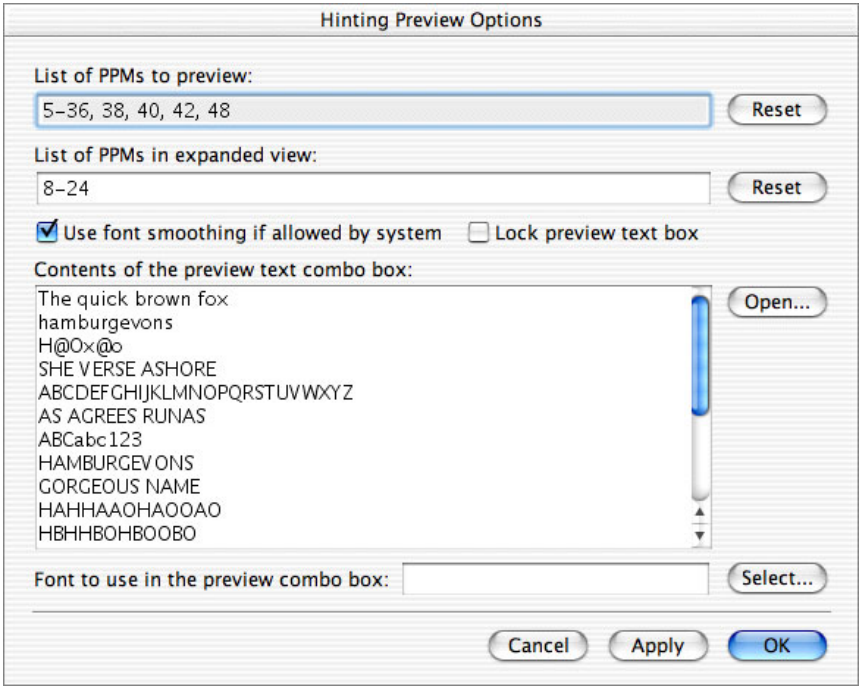
Type 1 Preview
The quick brown fox
...
The quick brown fox
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
8 The quick brown fox
9 The quick brown fox
10 The quick brown fox
11 The quick brown fox
12 The quick brown fox
13 The quick brown fox
14 The quick brown fox
15 The quick brown fox
16 The quick brown fox
17 The quick brown fox
18 The quick brown fox

The top part of the panel remains the same, and the bottom part is the sample string rendered at multiple PPMs. You close the panel by clicking the button.

Preview Panel Options

With the Preview panel options dialog box you can customize the PPMs at which the horizontal waterfall and sample string are rendered.

To open the dialog click on the ... button to the right of the sample string editing control. You will see the options dialog box:



At the top of the dialog box you will find two editing fields where you can enter the **list of PPMs to preview** in the horizontal waterfall line (top control) and in which to preview the sample string in the expanded mode (bottom line).

Enter the PPMs separated by commas or define ranges of PPMs using the '-':

```
12, 13, 15, 16-24
```

Click on the **Reset** button to the right of the editing fields to reset the list of PPMs to the default values.

Below the PPM selection controls there are two check boxes:

Use font smoothing if allowed by the system
Check this control to force the system rasterizer to preview your font with smoothing. This option is off by default while smoothing in Mac OS is on by default.

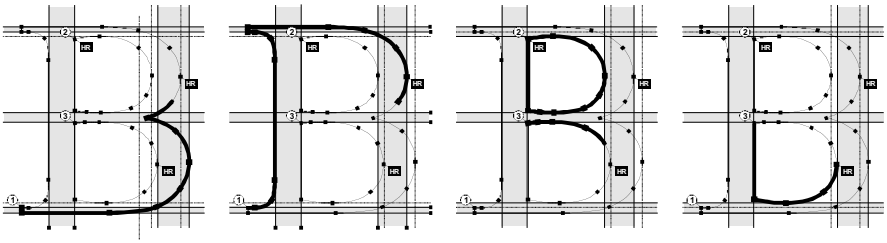
Lock preview text box
When this option is switched on, FontLab will not allow you to change the contents of the sample string edit box. You can, however, select pre-defined strings in the list.

Below the options there is a **list of pre-defined sample stings**. You can freely edit it as text or click the **Open** button to open a text file that will be used as a source of sample strings.

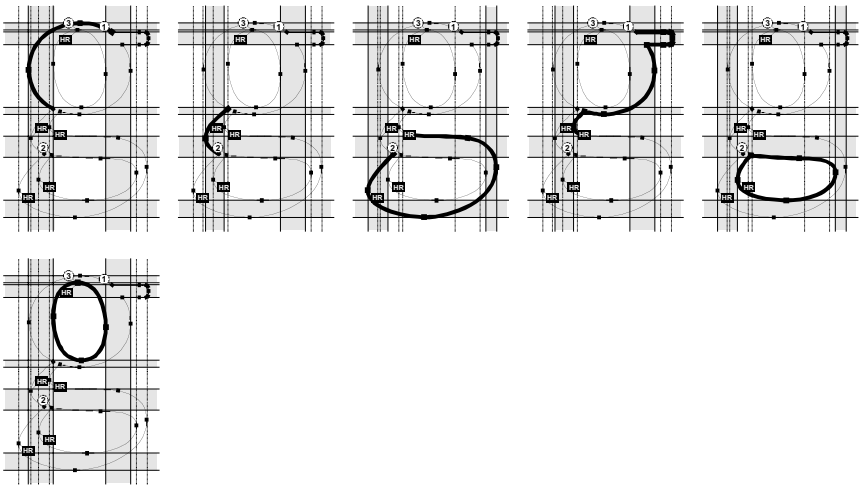
The last option in the dialog box is the **Font to use in the preview combo box**. Enter the font name or click on the **Select** button to choose one of the fonts. The font will be used in the sample string selection control – this is very useful when you are working with non-Latin fonts.

Some Examples

Here we want to show you some typical hinting situations and recommended hinting sequences for them:



Example of a hint replacement program for the Times 'B' character



Example of a hint replacement program for the Times 'g' character

TrueType Instructions

In TrueType fonts, the hinting process is very different from the one used in Type 1 fonts. As we said before, in TrueType font format almost all characters have special programs that directly control the movement of outline points at different PPM sizes.

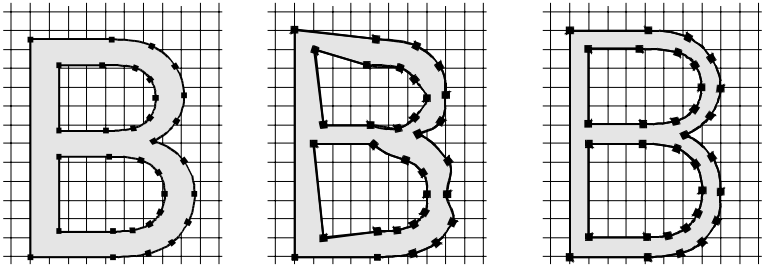
The native TrueType instruction language consists of several dozen commands. All the commands deal with the data stack, a temporary storage place, and the constant definitions that come with a font.

There are 3 different kinds of instructions in each TrueType font file. One global program (called the Font Program) is executed one time when the font is used for the first time. Another global program (called the PPM Program) is executed one time when the font's PPM is changed. Local programs (Glyph Programs) are executed for each glyph when it is scaled.

Programs can deal with points, distances, arithmetic values, constants and graphics state parameters. Graphics state parameters set rules that are used as settings for various commands.

Every character outline is scaled according to the selected PPM value. Point coordinates are stored as fixed-point numbers (they are not integers, but have a fixed precision). Then the glyph program is interpreted. The glyph program measures the distances between outline points, uses font-level constants and resets the position of some outline points. These points are called *touched* points.

All untouched points usually interpolate (by the last glyph program command) between the new positions of touched points.



Original outline

Hinted outline before interpolation

Hinted outline after final interpolation

There are several special commands that are applied to the glyph outline at specific PPMs. These commands are called *delta instructions* and are used to slightly modify the position of outline points to improve the character's appearance.

We will not include here a complete description of all TrueType instructions. If you are interested we recommend that you read "Technical Specifications for TrueType Font Files" which is available from Microsoft.

Font Parameters

Because within each font it's very important that all characters have a consistent appearance, some font-level information is necessary for hinting. There are two kinds of such information: alignment zones and stem widths.

Alignment zones set the positions of the most important vertical positions, such as the sizes of uppercase characters, the position of middle-lines, and the top and bottom overshoot positions.

To minimize rounding error and to make better baseline alignment you have to suppress overshoots and maintain the same size of overshoot and regular characters at small PPMs. This is very easy to do if you position and size the bottoms and tops of these characters and "stick" the bottom and top points to these values.

Stem widths define the most important stems in the font and control the rounding of these stems. Here you must control the PPM at which the stem width changes from one to two pixels and from two to three pixels.

Refer to page 486 for more information about alignment zones and standard stem widths.

Visual TrueType Hints

In FontLab we use a small set of high-level hinting instructions that are automatically compiled to TrueType instructions during font export. Because these instructions can be set and edited visually we call them *visual TrueType hints* or just *visual hints*.

Visual hints are enough to define TrueType hints even in very complex situations and they are compiled in very compact and effective TrueType instruction code.

The visual hint set includes the following commands:

Align	Aligns (moves) the position of the outline point to the designated position on the grid or to the edge of the alignment zone.
Single Link	Sets the position of the point relative to the position of another point. Distance can be linked with one of the stem widths. Distances also may be rounded or not.
Double Link	Sets the distance between two points to an integer value that may be linked with a stem width
Interpolate	Interpolates the position of a point between two other points
Middle Delta	Slightly shifts a point at a specific PPM. This command works *before* the final interpolation of untouched points.
Final Delta	Slightly shifts a point at a specific PPM. This command works *after* the final interpolation of untouched points. This command is used for the final outline correction.

All commands are available in horizontal and vertical directions. There are no “diagonal” visual instructions.

Sequence of commands

Visual commands may be set in any sequence, but they are interpreted in a very specific order. FontLab automatically detects the logic of the hinting program and does intelligent sorting.

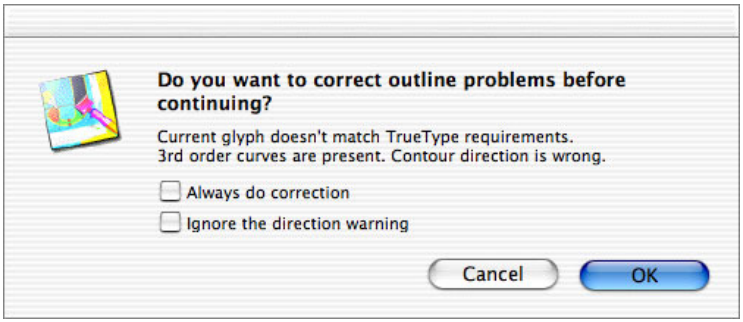
1. Align commands are always interpreted first.
2. Double links are interpreted first also, except that middle delta instructions may set points that are linked by double links.
3. Single links and interpolate commands are interpreted in logical sequence.
4. Middle delta commands are interpreted after commands that set positions of the points for which they are set, but before commands that are based on these points.
5. Final delta commands are interpreted after final interpolation of the untouched points.

TrueType Hinting Tool

With the TrueType hinting tool you can set and modify visual TrueType hints and preview the real resulting TrueType font using the system TrueType rasterizer.

To activate the TrueType Hinting Tool select the **TrueType Hinting** command from the **Tools/Operations** menu or press the button on the Tools or Operations toolbar.

When you select the TT Hinting Tool you may see the message:

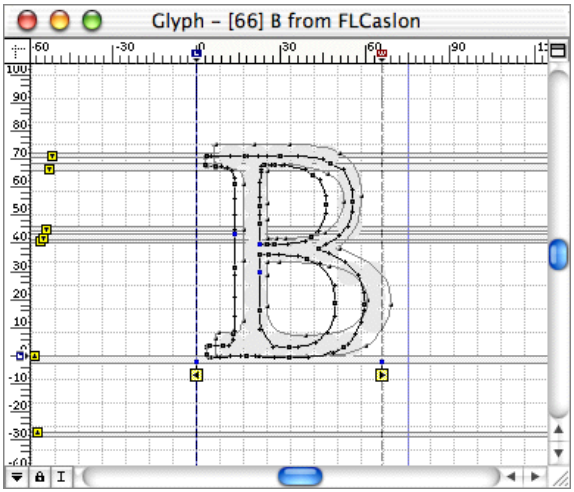

Do you want to correct outline problems before continuing?
Current glyph doesn't match TrueType requirements.
3rd order curves are present. Contour direction is wrong.
Always do correction
Ignore the direction warning
Cancel
OK

This message appears if the current character has 3rd-order curves that should be converted to 2nd-order curves to set Visual TrueType hints or if the contour direction is wrong.

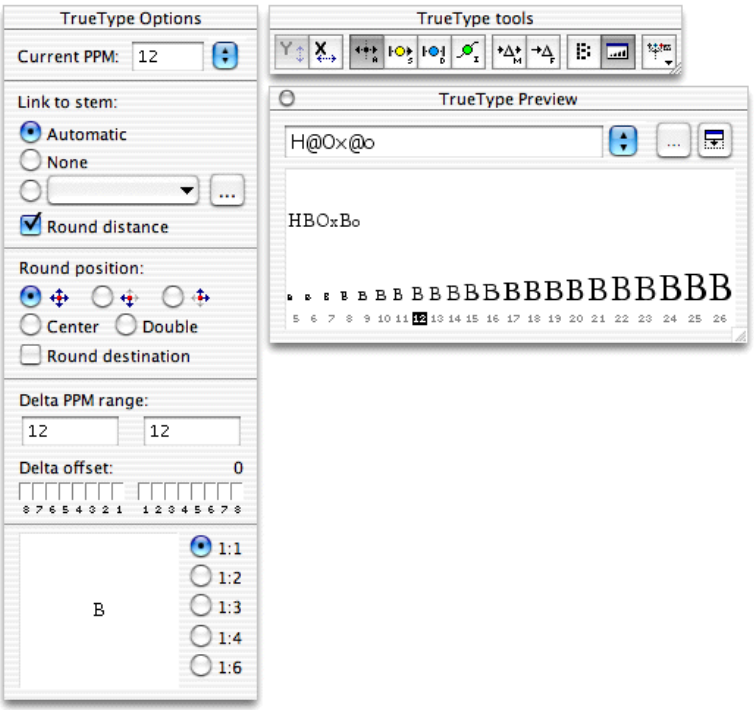
The TrueType specification requires that contours in TrueType fonts to be directed clockwise. Most rasterizers will correctly render TrueType fonts with incorrectly directed contours, but it is not guaranteed so we recommend directing contours according to the specification. If you don't want FontLab to check the contour direction, check the **Ignore the direction warning** option.

Press the **OK** button to continue to work with the tool. Check the **Always do correction** option if you want FontLab to always correct the outline format when opening a character.

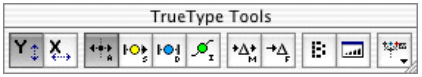
When the Hinting Tool becomes active you will see that the editing field of the Glyph Window has changed:



The TrueType toolbar, Options, and Preview may appear on the screen:



TrueType Toolbar

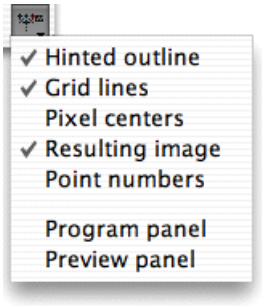


With the TrueType Toolbar you can select the hinting direction, the current visual instruction tool, and the layers that appear in the editing field of the Glyph Window. You can also open and close the Preview and Program panels.

Here is a detailed description of the TrueType Toolbar buttons:

	Set vertical direction. Sets visual instruction tools to work in the vertical direction. I.e. create vertical links, alignment, interpolation and deltas. Instructions in different directions are independent of each other.
	Set horizontal direction
	Select Align command
	Select Single Link command
	Select Double Link command
	Select Interpolate command
	Select Middle Delta command
	Select Final Delta command
	Switch TrueType hinting tool to bitmap mode
	Show Preview panel
	Open the Options menu.

If you click on the button, you will see the Options menu:

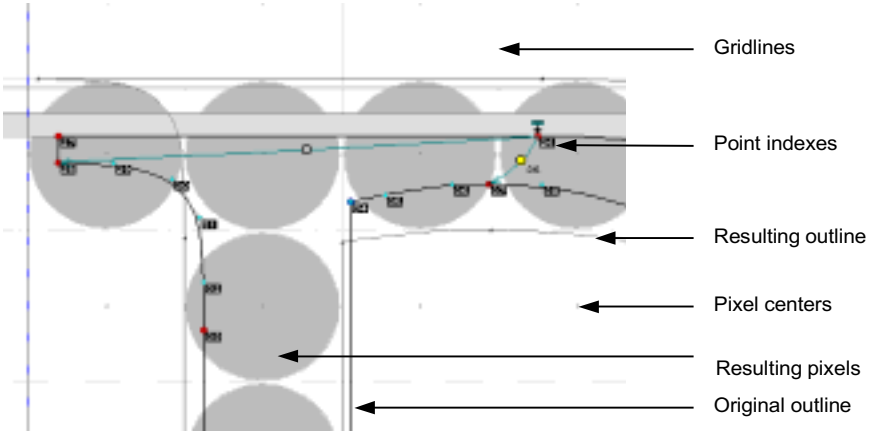


All the commands except the two at the bottom control the appearance of the layers that represent information about the hinting process. The layers are described in the following section.

The two remaining commands let you open the Programs panel that previews the source code of the hinting program and the Preview panel that shows the result of the hinting.

Layers

With the TrueType tool you can see various information layers in the Glyph Window.

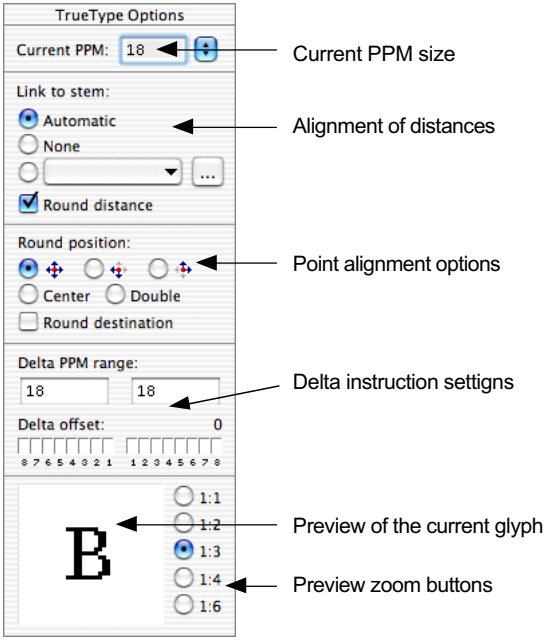


These layers are different from the usual editing layers, so we will explain them here:

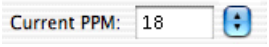
Outline	The original, untouched outline of the glyph always appears in the Glyph Window in black. The Resulting outline, which is the result of the interpretation of the instruction program, appears in gray.
Grid	Gridlines. Gridlines mark the edges of pixels that will appear in the selected PPM size.
Centers	Centers of pixels. When the outline is filled, all pixels whose centers are on the outline or inside the outline are set black. This layer is very useful for delta hinting.
Pixels	Resulting pixels. FontLab uses the real TrueType rasterizer to preview a filled glyph at the selected PPM after interpretation of the instruction program.
Point indices	Point numbers. There are also two special points that mark the left and right sidebearings of the character. You can apply any commands to any of these points.

Options Panel

The main control center of the TrueType hinting tool is the Options panel:



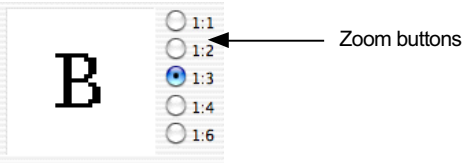
The main use of this panel is for selection of the current PPM size. Use the drop-down combo box to select one of the predefined values or enter your own value in the edit field:



You can also browse the predefined PPM values by pressing the **CMD-PGUP** and **CMD-PGDN** keys on the keyboard.

In the options preview window you can see a sample of the current character. FontLab uses a real Mac OS TrueType rasterizer to render the character, so this preview is exactly what you will see when you export a TrueType font and use it in an application in a black-white mode (without smoothing).

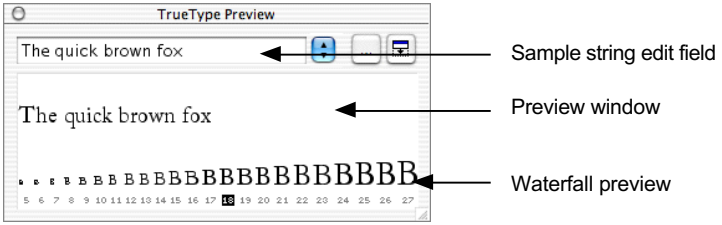
Use the zoom selection buttons to enlarge the sample character:



All other options that can be set in this panel will be described later.

Preview Panel

In the preview panel you will see a waterfall preview of the current character and a preview of a string at the current PPM size:


TrueType Preview
The quick brown fox
The quick brown fox
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
Sample string edit field
Preview window
Waterfall preview

To open the Preview panel (if it's not already open) press the button on the TrueType toolbar.

If you enter any characters in the edit field, you will see these characters appear in the top of the preview window. Use the usual FontLab rules to enter special characters (enter the character name after a slash '/' character or two slashes to enter a slash: "//").

You may also drag selected characters from the Font Window and drop them in the Preview window.

With the waterfall preview you can quickly select the current PPM. Just move the cursor to the sample PPM you want to select and double-click the mouse. You can also browse the PPM values by pressing the **CMD-PGUP** and **CMD-PGDN** shortcuts on the keyboard.

How FontLab Previews TrueType Fonts

FontLab uses the real Mac OS TrueType rasterizer(ATSUI) to give an exact preview of the TrueType hinting for the Preview panel, the Pixels layer in the Glyph Window and the preview window in the Options panel. To get a preview FontLab implements the following steps:

1. FontLab creates a very small TrueType font that includes the empty (notdef) character, current character and all characters from the sample string of the Preview panel.
2. All characters in this font are mapped to codes starting from the 0x3F (63) – code of the question character. The empty character is mapped to 0, as usual.
3. This font is exported inPC TrueType format (.ttf) using all the TrueType export settings, including grayscale settings, maintaining the existing instructions setting and autohinting. A special font name is set for this font to avoid conflict with any installed fonts.
4. This font is saved in the temporary directory and installed in the system.
5. All previews are made with this font.
6. When the font is no longer needed it is uninstalled and the font file is removed.

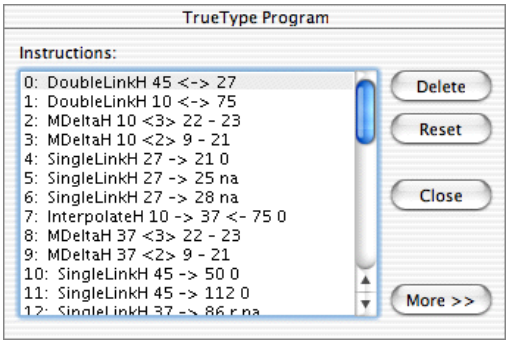
Like the Type 1 hinting preview panel, the TrueType hinting preview panel can be opened and its options can be customized. Please refer to the description of the Type 1 hinting preview panel on page 504 for more information.

Font Smoothing

Use the **Use font smoothing if allowed by system** checkbox in the options dialog to switch the smoothing on and off. Mac OS X and Classic render text in different ways. To control the system smoothing in Mac OS X, use the General page of the System Preferences. In Classic use the Fonts page of the Appearance control panel.

Program Panel

In the Program panel you can see the text of the instruction program as it is interpreted:


TrueType Program
Instructions:
0: DoubleLinkH 45 <-> 27
1: DoubleLinkH 10 <-> 75
2: MDeltaH 10 <3> 22 - 23
3: MDeltaH 10 <2> 9 - 21
4: SingleLinkH 27 -> 21 0
5: SingleLinkH 27 -> 25 na
6: SingleLinkH 27 -> 28 na
7: InterpolateH 10 -> 37 <- 75 0
8: MDeltaH 37 <3> 22 - 23
9: MDeltaH 37 <2> 9 - 21
10: SingleLinkH 45 -> 50 0
11: SingleLinkH 45 -> 112 0
Delete
Reset
Close
More >>

To open the Program panel use the command from the TrueType hinting options menu.

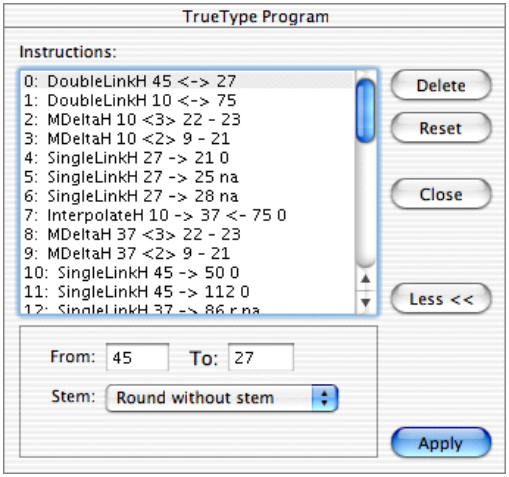
The Program panel consists of the program listing and four buttons:

Delete	Removes the selected command
Reset	Removes all commands
Close	Closes the panel
More>>	Expands the panel to include the command options controls.

Here are the mnemonics that are used to represent commands. A detailed description of each command and its parameters will follow.

AlignTop	Align a point to the top alignment zone
AlignBottom	Align a point to the bottom alignment zone
Align	Align a point to the grid
SingleLinkH, SingleLinkV	Single Link commands in horizontal and vertical directions
DoubleLinkH, DoubleLinkV	Double Link commands in horizontal and vertical directions
InterpolateH, InterpolateV	Interpolate commands in horizontal and vertical directions
MDeltaH, MDeltaV	Middle Delta commands
FDeltaH, FDeltaV	Final Delta commands

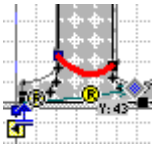
Click on the **More>>** button to expand the panel:


TrueType Program
Instructions:
0: DoubleLinkH 45 <-> 27
1: DoubleLinkH 10 <-> 75
2: MDeltaH 10 <3> 22 - 23
3: MDeltaH 10 <2> 9 - 21
4: SingleLinkH 27 -> 21 0
5: SingleLinkH 27 -> 25 na
6: SingleLinkH 27 -> 28 na
7: InterpolateH 10 -> 37 <- 75 0
8: MDeltaH 37 <3> 22 - 23
9: MDeltaH 37 <2> 9 - 21
10: SingleLinkH 45 -> 50 0
11: SingleLinkH 45 -> 112 0
Delete
Reset
Close
Less <<
From: 45
To: 27
Stem: Round without stem
Apply

In expanded mode you see the command options panel in the bottom. Select the command and use the controls in that area to change the command parameters. Click the **Apply** button after you make changes to accept them.

The commands in the list have a two-way connection with the Glyph window: when you click on the visual representation of the command in the Glyph window, the related command is selected in the Program panel's list.

When you select a command in the Program list, the command is visually highlighted in the Glyph window:



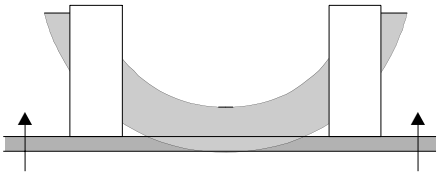
Use the command link to check the logic of the hinting program and to customize command parameters in complex situations.

Alignment Instructions

Alignment instructions are used to align points to the grid. There are two types of alignment instructions: those linked with alignment zones and those not linked with zones.

Alignment Zones

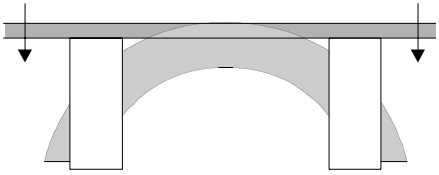
Alignment zones define important vertical positions that are common to many font characters. A good example of an alignment zone is a baseline and the bottom position of the 'O' character:



Bottom alignment zone

At low PPM sizes you must set the bottom point of the characters 'O', 'C' and similar characters to the baseline to suppress the bottom overshoot. In this case we have a so-called bottom alignment zone.

Another example of an alignment zone is the top line of the 'H' character and the 'O' character:



Top alignment zone

Alignment zones are previewed in the Glyph Window with a gray color.

You can set and edit alignment zones in the **Alignment Zones** page of the Font Info dialog box. See page 196 for a detailed description of this page.

TrueType alignment zones are the same as those used in Type 1 font-level hinting.

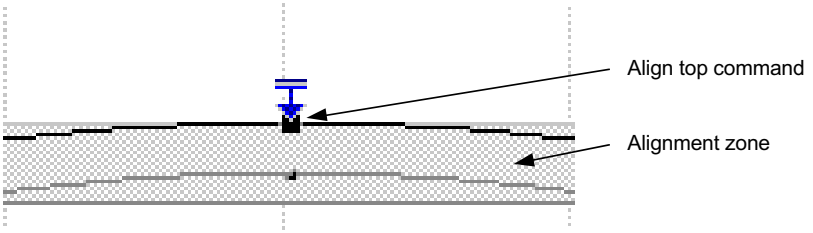
AlignTop and AlignBottom Instructions

These instructions are available only when the vertical hinting direction is selected.

To add AlignTop or AlignBottom commands:

1. Select the Align tool .
2. Position the cursor over the point that you want to align and click the mouse button
3. If the selected point was in one of the alignment zones, an AlignTop or AlignBottom command will be added to the hinting program. If the selected point is not in the zone, the “free” Align command will be added.

The AlignTop command appears as:



The AlignBottom command is similar to the AlignTop command.

How AlignTop and AlignBottom Commands Work

1. In the prep program (this program interprets every time the PPM is changed) all alignment zones (stored in the cvt table) are aligned to the closest integer position.
2. When the glyph program interprets, the position of the point aligned to the zone by the AlignTop or AlignBottom command is set equal to the aligned position of the zone if the scaled distance between it and the zone is no more than 16/17 pixels.

The AlignTop and AlignBottom commands have two arguments: the index of the point that is aligned by the command and the index of the zone. In the Program panel these commands appear as:

AlignBottom 12 [1]
AlignTop 5[0]

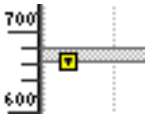
where 12 and 5 are point indexes and 1 and 0 are indexes of the bottom and top zone, respectively.

Hinting Alignment Zones

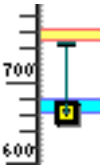
By default the position of the alignment zones is scaled linearly. Sometimes this may not be satisfactory and you may need to exactly specify the position of the alignment zones at some PPMs.

With FontLab you can adjust the position of the zones around the linearly scaled position.

Every alignment zone has a yellow icon at the left:



You can drag this icon up or down to adjust the position of the zone at the current PPM:



If you move the icon you will see all glyphs that have align commands applied to that zone changed their shape.

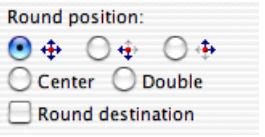
You may find this feature useful when you have to add glyphs to an already hinted TrueType font. New glyphs often have a different height at some PPMs and you can correct that with the zone alignment command and zone hinting.

Align Instruction

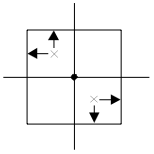
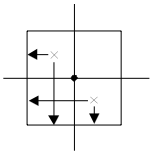
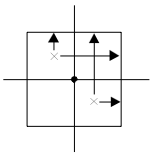
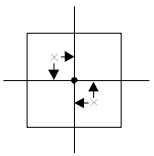
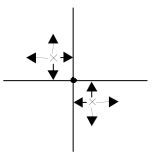
Use the Align command when it's not necessary (or possible – you can't align points to horizontal alignment zones) to align a point to an alignment zone but you want to align the point's position to the grid. The Align command is available in both directions.

The Align command allows you to directly control how point coordinates are rounded.

To select a rounding method use the alignment control radio buttons in the Options panel:



Here is a graphical description of the various rounding methods:

Rounding	Appearance	Code	Description
		0	Aligns to the closest grid line
		1	Aligns to the left or bottom grid line
		2	Aligns to the right or top grid line
		3	Aligns to the closest center of pixel
			Aligns to closest edge of pixel or center of pixel

To set an alignment command on the outline:

1. Press the align button on the toolbar.
2. Position the mouse cursor on the point that you want to align.
3. Click the mouse button. Hold the **SHIFT** key if you are setting vertical commands and the point is inside an alignment zone.

In the Program panel the Align command appears as:

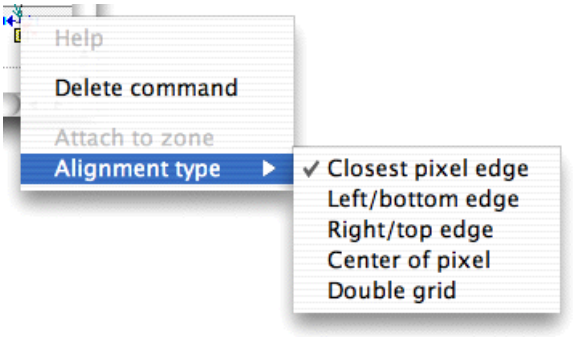
AlignV 12 [1]
AlignH 22 [0]

where 12 and 22 are point indices and 1 and 0 are the codes of the alignment rounding method as shown in the table above.

Align commands are the very first commands in any hinting program. They do not have base points and affect only one point.

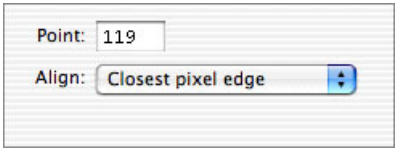
Customizing the Align Instructions

You can customize Align instructions with the context menu. **Ctrl**-click the point to which the align command is to be applied and you will see the menu:



You may delete the command, attach or detach it from the alignment zone (this option is available only if the point is in the zone) and change the alignment type.

Another way to customize the command is to use the Program panel in expanded mode:



Links

Links are the most important visual commands. They are used to set relationships between outline points and to set distances between points to one of the standard stem values.

There are two types of links: single links and double links. Single links need to have a base point that is set by a previous command. This can be any command except a final delta.

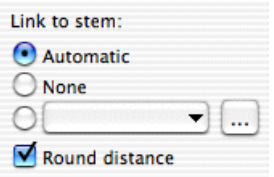
A double link does not need to have base points. It sets the position of two points and they can be used as base points for a single link or interpolate command.

Standard Stems

All link commands can be connected with one of the standard stems. When a link is connected to a stem it sets the distance between points to the scaled and rounded width of the stem. Using this technique you can implement standard-stem-based hinting, keeping important stem widths the same in every character in which they are used.

Please note that Type 1 standard stems and TrueType stems are different, although they function similarly. There is no limitation to the number of TrueType stems and they may be named. We will discuss all the different options of TrueType stems later on page 553.

You can use the Options panel **to select standard stem options**:



There are three main options: **Automatic** connection to the stem, **Manual** connection and no connection at all.

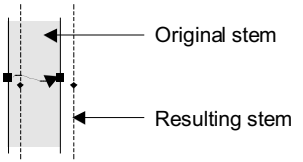
In automatic mode FontLab will automatically select the standard stem that is closest to the linking distance. It will not connect the link with the stem if the difference between the distance and the stem width is more than either the stem or the distance.

In manual mode, you select a stem to which you want to connect a link in the stems popup menu and switch on the radio button to the left. However, if the difference between the distance and the selected stem is too high, FontLab will not connect the link to the stem.

To keep from connecting a link with a standard stem switch on the **None** option.

Single Links

The single link connects an outline point to a point whose position is set by another command. If you set a single link that linked point will always be at the given distance from the base point:



Single link instruction

A single link may be linked with one of the standard stem widths. If it is so linked then the distance to the base point is replaced by the scaled and rounded stem width. So if several links are connected to the same stem width the distances from the base points will always be the same.

Single links are very straightforward: they have one base point and one affected point.

If a single link is connected with one of the standard stems the linking distance is always rounded to the grid. If the link is not so connected, then it may be rounded or not depending on the currently selected **Round distance** option:

Rounding distances is the default option for this command because it usually gives better control of the position of the destination point.

There is a special kind of single link called the *aligned single link*. It can be described as a combination of a single link and the Align command on the linked point. You can't set a normal Align command at the end of the link but you can use an aligned link in this case. To make a link aligned switch on the **Round destination** option on the Options panel:

Aligned single links use the current Align mode.

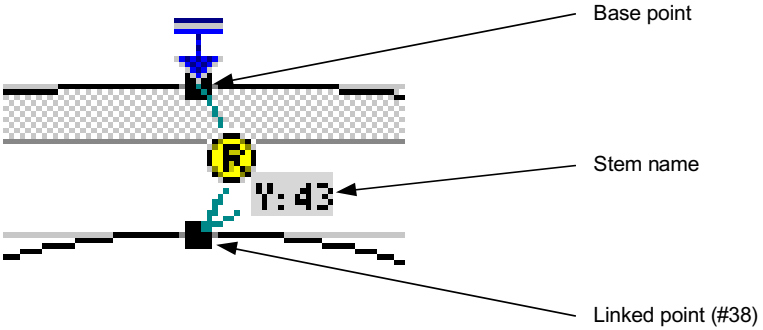
To set a single link:

1. Select the single link tool .
2. Position the cursor on the base point. Don't forget that the position of the base point must be set by an instruction.
3. Press the mouse button and drag the mouse to the point that you want to link. A circle will highlight it. Release the button.

Hold the **Shift** key down if you do not want to connect the single link to a standard stem width. If you hold the **Shift** key the stem settings in the Options panel will not work.

If the base point of the link is not aligned by some other command, FontLab will automatically set an Align command on it. Hold down the **Cmd** key if you want to prevent this.

In the Glyph Window the single link command appears as a directed line with a mark in the middle. The mark is yellow if the link is connected to a standard stem width, and gray if it is not. If the link is connected, the stem name may appear near the mark:



In the Program panel a single link command appears as:

SingleLinkV 12 -> 38 [0] 1

where 12 is the base point; 38 is the linked point; 0 is the number of the stem to which this link is connected (or “ns” if it is not connected to the stem); and 1 is the type of alignment for aligned links (or “na” for links that are not aligned).

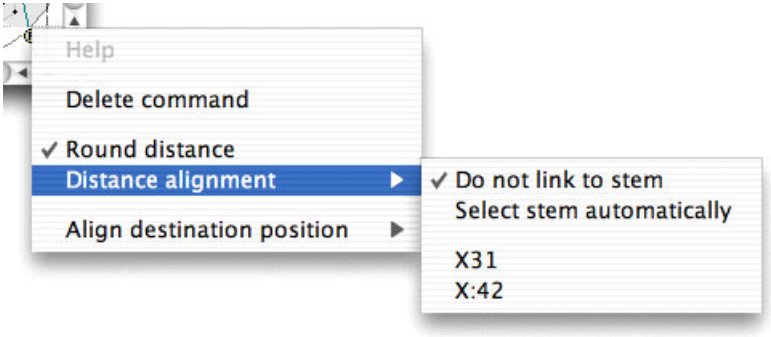
How the SingleLink Command Works

1. First the original (not grid-fitted) positions of the base point and the linked point are retrieved and the distance between them is measured.
2. If the link is connected with a stem the distance is replaced by the prepared stem width for this PPM. If the scaled distance is equal to or exceeds 3 pixels it is rounded to the closest integer. If the link is not connected to the stem the distance is rounded or not depending on the rounding option of the command.
3. The distance of the linked point is set equal to the grid-fitted position of the base point shifted by the distance.
4. If a link is aligned then the position of the linked point is rounded to the grid according to the align mode stored as the last command parameter.

Customizing the Single Link Command

There are two ways to customize the single link command: the context menu and the Program panel.

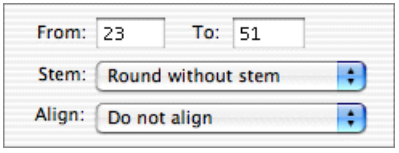
To customize a command with the context menu **Ctrl**-click the round icon in the middle of the link:



Use the **Delete** command to remove the command and the other options to customize it. The **Distance alignment** sub-menu lets you link the command to one of the stems.

The **Align destination position** submenu is the same that you can see in the Align command's context menu and defines the rounding of the destination position.

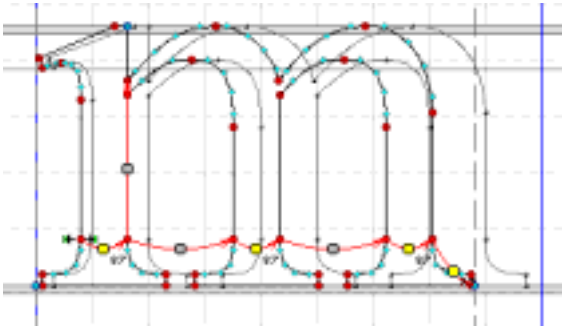
In the Program panel the Single link options looks like:



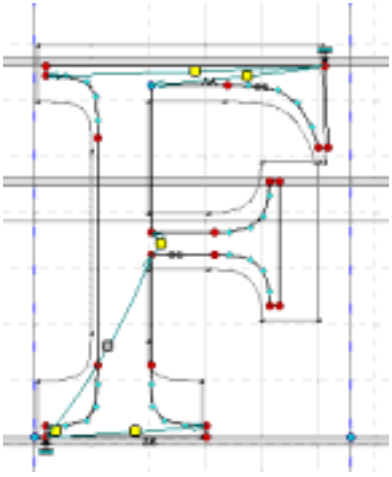
You can change the indexes of the starting and destination points and change the distance rounding and destination alignment options.

Single Link Examples

Here are some examples of single link hinting:



Horizontal hinting of the Times 'm' character

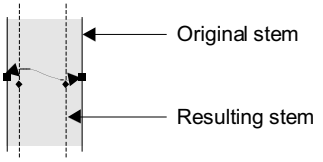


Vertical hinting of the Times 'F' character

Double Links

Double links let you set the distance between two “untouched” points. Both points will be moved by this command and both points will be touched afterward.

The distance between points can be connected to one of the standard stem widths using the usual procedure described above. If the distance is connected to the stem width, it will scale with the stem.



Double link instruction

You can’t predict where points will move that are connected by a double link so we recommend that you use this command only for stems for which the position is not very important or for stems that will be used as the basis for hinting.

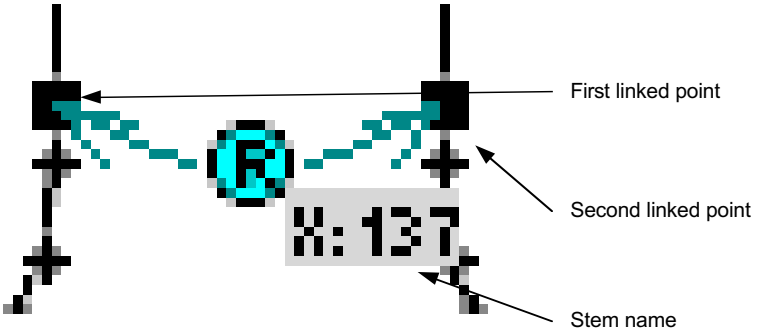
The double link command does not have base points, but it sets the position of points.

To set a double link:

1. Select the double link tool .
2. Position the cursor over one of the points you want to link. Press the mouse button.
3. Drag the cursor to the point you want to link to the first point. A circle will highlight any point over which you move the cursor. Release the mouse button when the desired point is highlighted.

Hold the **Shift** key down if you do not want to connect the double link to a standard stem width. If you hold down the **Shift** key the stem settings in the Options panel will not work.

In the Glyph Window the Double link command appears as a line with two arrows and a mark in the middle. The mark is blue if the link is connected to the stem, and gray if it is not connected. If the link is connected the stem width will appear near the mark:



In the Program panel a double link command appears as:

DoubleLinkV 12 <-> 36 [0]

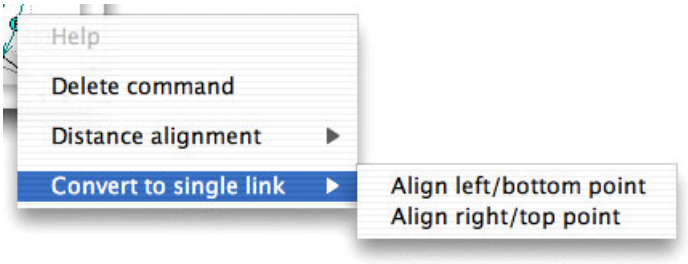
where 12 and 36 are points that are linked; and 0 is the number of the stem to which this link is connected (or "ns" if it is not connected to the stem).

How the Double Link Command Works

1. The original (not grid-fitted) positions of two linked points are retrieved and the distance between them measured.
2. If the link is connected with the stem the distance is replaced by the prepared stem width for the PPM. If the scaled distance is equal to or exceeds 3 pixels it is rounded to the closest integer. If the link is not connected to a stem the distance is rounded to the closest integer, starting from 1.
3. The grid-fitted positions of the two linked points are stored.
4. Both points are aligned to the closest grid line.
5. Rounding errors for both points are calculated by comparing original and rounded positions.
6. The point whose rounding error is less remains in place and the second point is set at the distance calculated in step 1. If the rounding errors are equal then the point with the lower coordinate value is set.
7. The distance of the linked point is set equal to the grid-fitted position of the base point shifted by the distance.

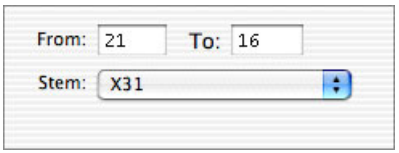
Customizing the Command

Ctrl-click the middle icon on the double link to open the context menu:



The **Delete** command and **Distance alignment** options are the same as in the **Single link** menu. An additional command is “Convert to single link”. Use it to replace the double link command with the combination of the align and single link commands.

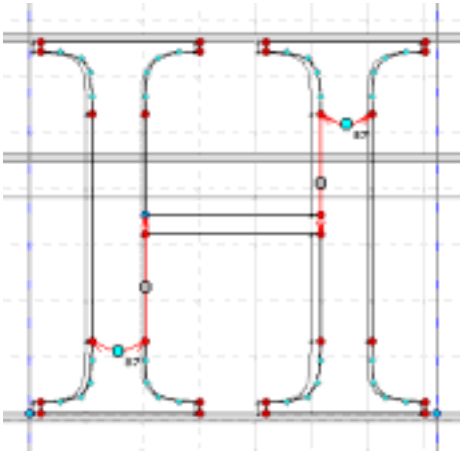
Program panel options for the double link command are simple:



There are controls to select the indexes of the linked points and the list of stem names to which the distance must be aligned.

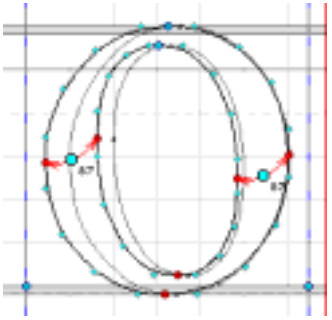
Double Link Examples

Some examples of double link command usage:



Horizontal hinting of the Times 'H' character

There are two double links and two single links in this example. The double links set the width and position of the two vertical stems and the single links align other points that are on the same stem.

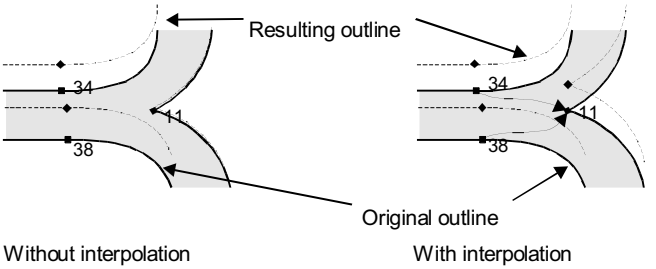


Horizontal hinting of the Times "o" character

Double links are usually used to hint characters in the horizontal directions in cases where stem position is not important. As you saw in the Single Links section we used single links to set horizontal instructions in the 'm' character because we wanted to keep the distances between vertical stems the same.

Interpolation

Most of the time, the align and linking commands are enough to hint most characters, but in some cases a very different kind of command is necessary. Look at the enlarged middle-right region of the character B in the vertical direction:



You can see that point 11 in the original outline in the vertical direction is set exactly between points 34 and 38. But because these points are connected by a double link and point 11 is not controlled it becomes too close to point 38.

In this and similar cases you can use the interpolate command. This command sets the position of a point in the same proportion between two other points as it was in the original outline.

Or the intermediate point could be aligned to he grid and serve as the starting point for a single link or another interpolation command.

This command has two base points and one affected point.

To set the interpolate command:

1. Select the interpolate tool .
2. Position the cursor over the first base point.
3. Holding the mouse button down move the cursor to the interpolated point and release the button.
4. Move the cursor to the second base point and click the mouse button.

In the Glyph Window the interpolate command appears as two directed lines from the interpolated point to base points.

In the Program panel the interpolate command appears as:

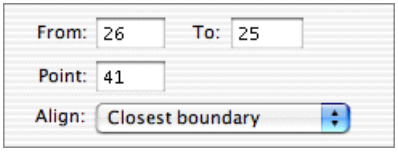
InterpolateV 34 -> 11 <- 38

where 11 is the interpolated point and 34 and 38 are the base points.

Customizing the Command

To customize the command **Ctrl**-click the command icon (small green circle). In the context menu you can use the **Delete** command to remove the command and the **Align destination** menu to align the destination point to the grid.

In the Programs panel the Interpolation command options look like this:



The Options panel allows you to choose the source and destination points for the command and to customize the alignment of the destination point.

Delta Instructions

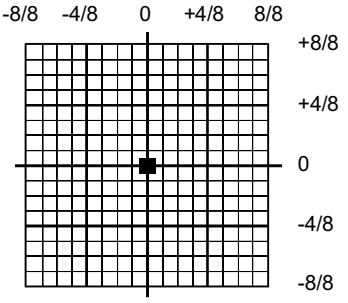
Now you know all about the visual hinting commands that can be applied to an outline at all PPMs. In addition to these commands the TrueType hinting language lets you set special commands that will work only at specific PPM sizes. These commands are called delta instructions.

There are two delta instructions for each hinting direction: middle delta instructions and final delta instructions.

Final delta instructions are necessary when you want to slightly change the instruction set – usually to add or remove some pixels:

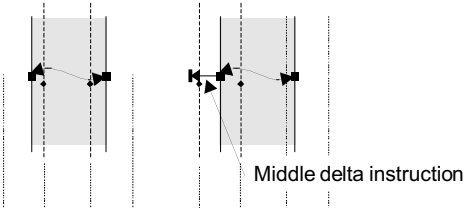
Middle Delta Instructions

A middle delta instruction can move outline points like any other instruction. It can move any outline point in 1/8-pixel increments from one pixel left (or bottom) to one pixel right (or top):



Movement space of delta instruction

Middle delta instructions are interpreted like other commands in the hinting program. They are automatically inserted between “normal” instructions that move points and so can be used to modify the normal interpretation of the hinting program:



In the example above, the vertical stem is moved one pixel to the left when the middle delta instruction is applied.

A middle delta instruction may be applied to a touched or untouched point and the point becomes touched afterwards.

It’s a good idea to use middle delta instructions to correct the rounding direction on some PPMs, like on the character ‘e’ at small PPM sizes.

Another good application of middle delta instructions is the correction of diagonal hints. A middle delta instruction “touches” the point to which it is applied, so any other point between two delta instructions will be interpolated in the final interpolation routine.

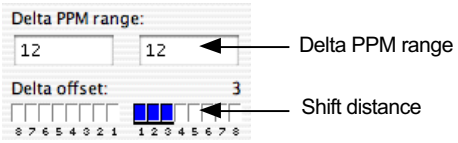
The middle delta instruction has 4 arguments: point to move, PPM_0, PPM_1 (the PPM range in which this instruction works), and the shift distance in eighths of pixels. In the Program panel this instruction looks like:

MDeltaV 18 <6> 12-14

where 18 is the point index, 6 is the distance to move (in this case – move up by 6/8 of a pixel. A minus sign in front of the number would indicate a move down), 12 is PPM_0, and 14 is PPM_1. This means that point 18 will be moved up 6/8 of a pixel when PPM is 12, 13 or 14.

To set a middle delta instruction:

1. Select the Middle Delta tool .
2. In the Options panel select the shift distance, direction and PPM range:



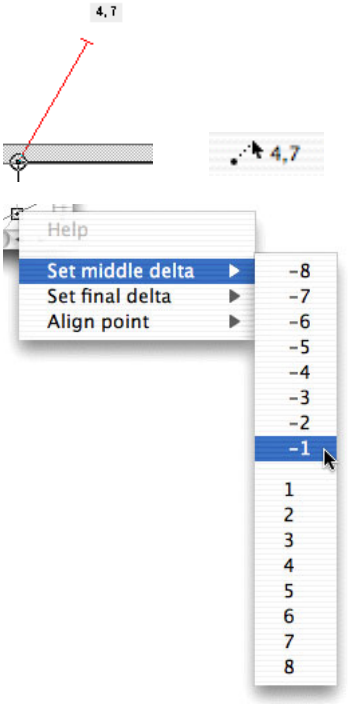
   By default the PMM range is equal to the current PPM.

3. Move the cursor to the point at which you want to set the delta instruction and click the mouse.

Faster Methods to Set Delta Instruction

There are two other methods to set delta instructions. The simplest method is to select the Middle Delta tool, position the mouse cursor over the point you want to move and drag the point in the needed direction. Hold down the **Shift** key to move the point only in the current hinting direction (horizontal or vertical).

While you are dragging the mouse to set the instruction you will see its value both near the end of the line and on the Meter bar:



You can also **Ctrl**-click any point on the outline and select the delta command in the context menu.

Active and Inactive Delta Instructions

Delta instructions may be active or inactive. When the current PPM is within the PPM range of the delta instruction the instruction is active. Otherwise it is inactive.

Active delta instruction (shift up)

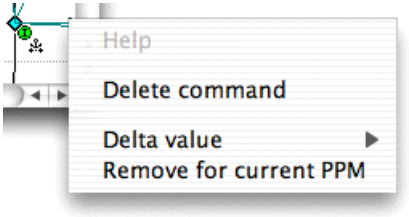
Inactive delta instruction

Note: if you set a new middle delta instruction at a point where a delta instruction is already set and is active the old instruction will be replaced by the new one.

If you set delta instructions for the same point and with the same shift distance but for different PPM ranges FontLab will try to combine PPM ranges and unite delta instructions where it is possible.

Customizing the Delta Instruction

Ctrl -click the delta instruction mark to get access to the context menu:

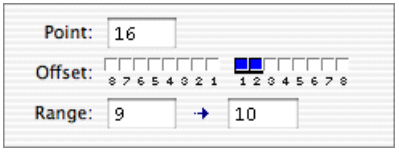


Use the **Delete** command to remove a middle delta instruction, so it will be removed for all PPMs for which it is defined.

Use the **Remove for current PPM** command to remove a middle delta instruction only for the currently selected PPM. For example, if the delta instruction is defined for the PPM range 12-18 and the current PPM is 14, after you use **Remove for current PPM** you will get two delta instructions: 12-13 and 15-18.

The Delta value submenu lets you choose the delta instruction shift distance.

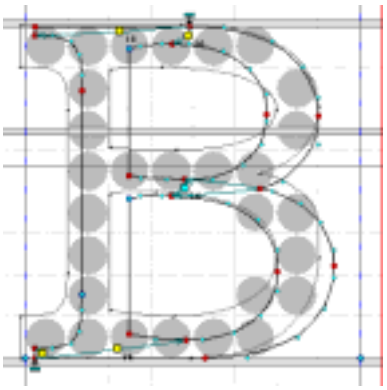
Another way to customize delta instructions is to use the options area in the Program panel:



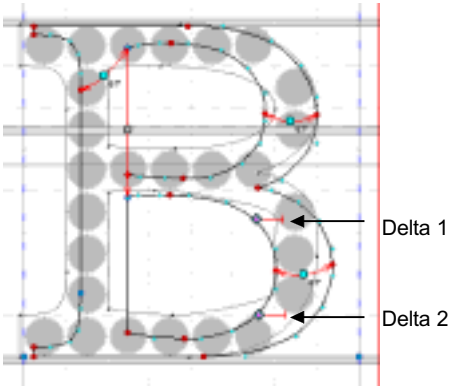
You can change the point number, shift distance and the PPM range.

Final Delta Instructions

Final delta instructions are applied after the final interpolation of all untouched points. They are used as a last resort to shift points to remove or add pixels at PPM sizes where it is still necessary after application of zones, stems and all other hints:



Hinted character at 12 PPM without delta instructions



Same character with two delta instructions

The sequence of interpretation is the only difference between middle and final delta instructions. They work exactly the same way. To set final delta instructions, select the final delta tool and follow the same instructions as above for middle deltas.

Some recommendations:

1. Delta hinting is a very time-consuming operation because you have to check the rendering results at many PPMs. So always try first to get the best possible results from normal, *linear* hinting.
2. Use the waterfall preview in the Preview panel to see where delta hinting is necessary.
3. Set up font smoothing in the General panel to see a real rendering preview. The General panel is accessible through System Preferences.
4. Try to set as few delta instructions as possible. They increase font size and complexity.
5. It's better to set one delta instruction with a larger range than two less comprehensive delta instructions.

Removing Instructions

You can remove any instruction using two methods:

1. Select the instruction in the Program panel and press the **Delete** button.
2. Position the cursor on the hotpoint of the instruction, hold down the **Ctrl** and **Cmd** keys and click the mouse.

   AlignTop, AlignBottom, Align, MiddleDelta and FinalDelta instructions have hotpoints on the point to which they are applied.

   SingleLink and DoubleLink instructions have hotpoints on the mark in the middle of the line that connects the two points.

   The Interpolate instruction has a hotpoint near the point that is interpolated.

Standard Stems

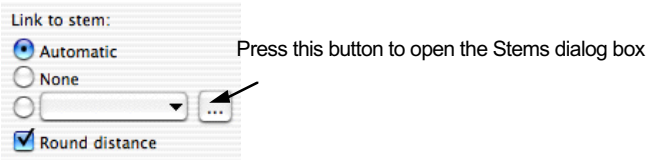
There are two sets of standard stems in FontLab: Type 1 standard stems and TrueType stems. Type 1 stems are used by the Type 1 font rasterizer and have some limits. For example, it is possible to define only 12 stems for each direction.

TrueType stems have no limits and also have some additional properties. They may be named and their rounding may be specified for selected PPMs.

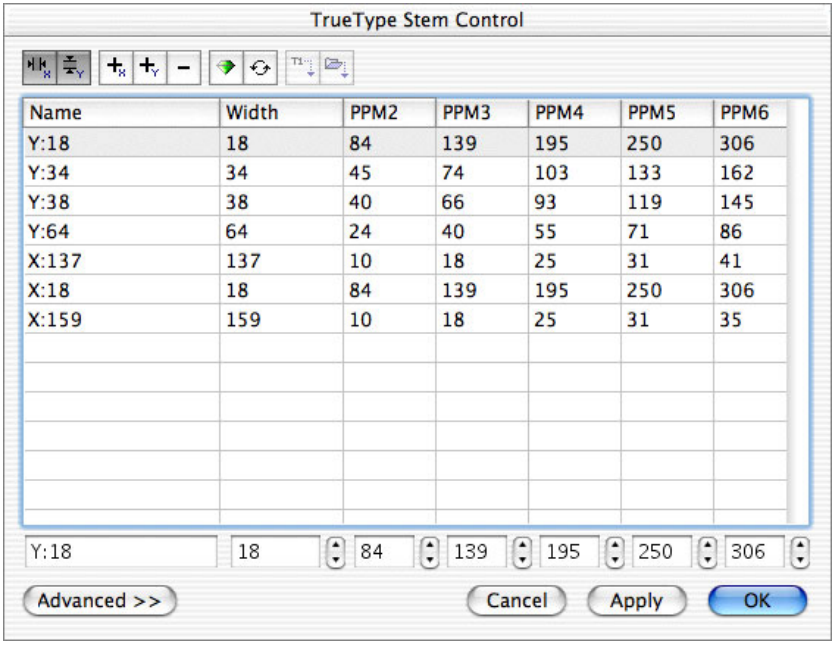
In FontLab you can control the rounding of the standard stem widths. You can set the PPM size at which any standard stem width is changed from 1 to 2 pixels, from 2 to 3 pixels, from 3 to 4 from 4 to 5 and from 5 to 6 pixels. We call these critical PPM sizes “jump” PPMs because here the stem width makes a jump from 1 to 2 pixels, from 2 to 3 pixels and so on up to 6 pixels wide. By default, standard stems are scaled linearly and are rounded to the closest integer value.

When you add a new Type 1 stem, it is added to both stem lists: Type 1 (editable with Font Info) and TrueType.

To control the TrueType stems' properties you open the TrueType stems dialog box:



You will see the following dialog box:



The dialog box consists of a toolbar on the top, a stem list, a row of controls to adjust stem parameters and the **Advanced** button that gives access to some font-level hinting options that we will discuss later.

Buttons on the toolbar mean:

	Include vertical stems in the list (stems measured along the X axis)
	Include horizontal stems in the list.
	Add new horizontal stem
	Add new vertical stem
	Remove the stem selected in the list.
	Automatically calculate and optimize the rounding PPMs.
	Reset rounding PPMs to linear values.
	Import Type 1 stems into the list.
	Import stems from another font.

Columns in the list mean:

Name	Name of the stem. You can use the control below the list to change the stem name. The stem name appears when the stem is used in hinting program. We do not recommend using long stem names, 6-8 characters is enough in most cases.
Width	Width of the stem. Use the control below the column to change the value.
PPM2-PPM6	PPMs at which the stem is scaled to 2-6 pixels.

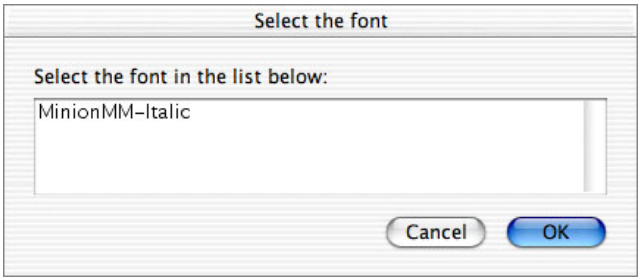
Click on the column header **to sort the list** by the column values. A yellow background in the list highlights vertical stems.

It is easy **to add a new stem**: just click on the or the button and a new stem will appear with the <unnamed> name and a width of 100.

To remove a stem, select it in the list and click on the button.

Click on the button to append Type 1 stems to the list. FontLab will append only those stems that are not already in the list.

With the button you can append stems from another font. Click it and select the source font from the list:



When you remove stems that are used by the hinting program in some glyphs FontLab will replace any stem-linked link with a simple rounded link.

Stem Rounding

Control over stem widths rounding is necessary for two reasons:

1. To control a font's contrast. If your font is low contrast, like Courier or Arial, you may want to set the jump PPMs for vertical and horizontal stems equal. This way, the font will never get high contrast, as when vertical stems are 2 pixels wide and horizontal stems are only 1 pixel wide.
2. To synchronise the scaling of stems with close widths – like the stems that control straight and round vertical stems in uppercase characters. At large PPM sizes they should be different but they have to make the 1→2 and 2→3 width jumps together.

Bigger jump values are necessary when you are working on a black or extra black font where 5-pixel width stems appear at low PPMs.

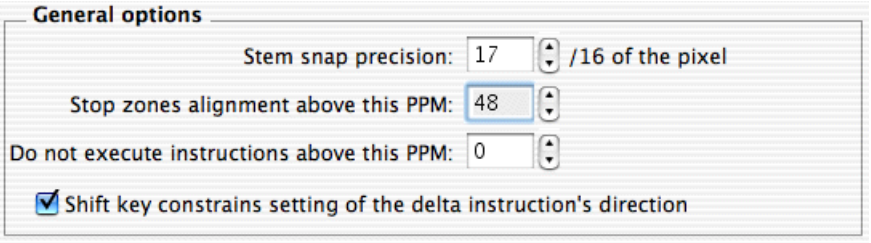
To change stem rounding:

1. In the list select the stem that you want to edit.
2. In the edit fields change the PPM sizes at which this stem makes its various jumps from one pixel up to 6 pixels.

Press the button to automatically optimize the stem rounding while trying to keep optimal contrast. Use the button to reset all stem PPMs to linear values.

Advanced Options

Click on the ... button in the Options panel and click on the **Advanced** button in the Stems dialog box. You will see a dialog box expand which includes the advanced options area:



Stem snap precision

Measured in 1/16 of a pixel, this value defines the difference between the actual stem width (scaled but not rounded) and the stem width in pixels specified by the stems dialog box. If the difference is more than the defined value, the stem is rounded linearly. A value of 16/16 will mean that stem rounding will be controllable only when the difference is less than one pixel. A value of 32/16 defines a possible difference of 2 pixels.

Stop zones alignment above this PPM

Enter the font size in PPM after which zone alignment will not be operative and all points controlled by the AlignTop and AlignBottom commands will be rounded to closest pixel edge.

Do not execute instructions above this PPM

Defines the PPM above which instructions will not be operative anymore. Enter 0 to never stop instructions.

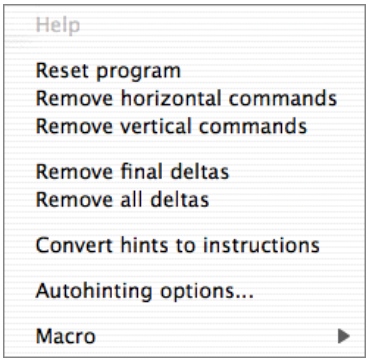
Shift key constrains setting of the delta instruction's direction

Inverts the function of the **Shift** key when you define delta instructions. By default you can freely move the destination point of the delta offset and you have to press **Shift** key to limit direction to the current hinting direction. When this option is deactivated, delta offset direction will be fixed and you have to press **Shift** key to "free" it.

You may click **Apply** button to immediately see results of the new options in the Preview panel.

Context Menu

If you **Ctrl**-click the mouse in the free space of the Glyph Window edit field you will see a popup menu:



The commands available in this menu are:

Reset Program	Removes all instructions
Remove Horizontal Commands	Removes all horizontal commands including delta instructions
Remove Vertical Commands	Removes all vertical commands including delta instructions
Remove Final Deltas	Removes all final delta instructions
Remove All Deltas	Removes all delta instructions (middle delta and final delta)
Convert Hints to Instructions	Tries to convert Type 1 hints and links to visual hints. You can use this command as a kind of TrueType autohinting. If a glyph doesn't have any Type 1 hints, use the Tools/Hints/Autohint command to generate them. Follow with the "Convert Hints..." command to generate a TrueType hinting program for the glyph
Autohinting options	Opens the Autohinting options dialog box that lets you customize the process of converting Type 1 hints to TrueType instructions. This dialog box is described below
Macro	Opens the list of macros available from the FontLab/Macro/System/TTH folder.

Autohinting Process

When you select the **Convert Hints to Instructions** command in the popup menu or apply the same action in the Transform or Transform Range dialog box, FontLab:

1. Converts all hints to links to find what points are linked by the hint.
2. For every horizontal link:

2.1 If one of the linked points is in the top or bottom alignment zone: adds an AlignTop or AlignBottom command and connects another point to it by the SingleLink command. Automatically links the SingleLink command with one of the standard stems.

2.2. If none of points is in the alignment zone: links them by a DoubleLink command or by a single link command if one point is linked to another outline point.

2.3 Checks all outline points and if one of them is on the same horizontal line as one of the linked points: links it by the SingleLink command. It doesn't do this if the point is adjacent to an already connected point.

3. For every vertical link:

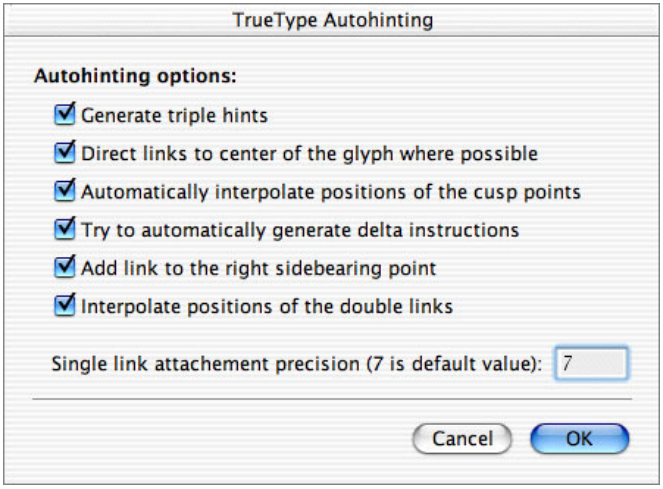
3.1 Connects two points by the DoubleLink command or by the single link command if one of the points is linked to another outline point.

3.2 Checks all outline points and if one of them is on the same horizontal line as one of the linked points: links it by the SingleLink command. It doesn't do this if the point is close to an already connected point.

4. FontLab optionally detects "cusp" points and links them to one of the single and double link commands.
5. For single and double link commands that connect round points, like on the left and right sides of the 'O' FontLab may optionally add middle delta commands that correct the rasterization of the curve.

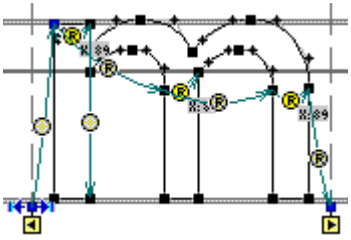
Autohinting Options

You can customize the conversion process with the **Autohinting Options** command. Use this command to open the dialog box:


TrueType Autohinting
Autohinting options:
Generate triple hints
Direct links to center of the glyph where possible
Automatically interpolate positions of the cusp points
Try to automatically generate delta instructions
Add link to the right sidebearing point
Interpolate positions of the double links
Single link attachement precision (7 is default value):
7
Cancel
OK

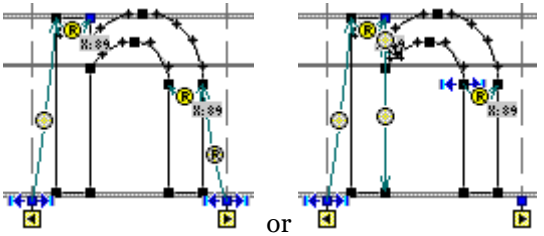
Generate triple hints

If this option is active FontLab will try to generate TrueType instructions that simulate Type 1 triple hints for characters like 'm' where the distance between stems must be kept the same at all PPMs:



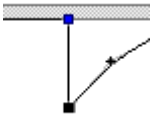
Direct links to center of the glyph where possible

FontLab can try to hint starting from the left sidebearing and going to the right or starting from both sidebearings and going to the center of the glyph. Use this option to customize the hinting direction:

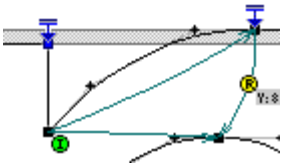

or

Automatically interpolate the position of cusp points

A Cusp is a point where the contour sharply changes direction:

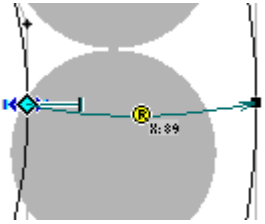


If this option is on, FontLab will generate an Interpolate command that will link the cusp point to one of the links:



Try to automatically generate delta instructions

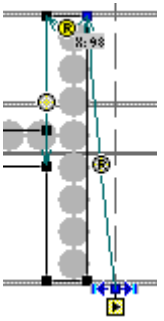
Our research shows that it is useful to include small middle delta commands at the end of links that connect round parts of the glyph:



Use this option to control this feature.

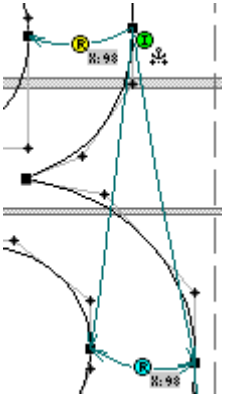
Add link to the right sidebearing point

FontLab can generate a single, distance-rounded link that will go from the rightmost hinted point to the right sidebearing point. This feature guarantees at least one pixel distance between glyphs on all PPMs.



Interpolate positions of the double links

In glyphs like ‘B’, which have overlapping stems, FontLab may hint them independently or hint the first stem and then interpolate the position of the second stem using the interpolation command with the final point aligned to the grid:



Whether this is appropriate for your font has to be decided on a case-by-case basis.

Single link attachment precision

If an outline is not very precise you can enter a value in this field so FontLab will have a threshold to decide when several points are located along a line. For precise fonts set this value to 1 or 0.

Generally speaking, autohinting is a trial and error process. You can get very good autohinting results if you select the correct options for your font and define correct stem and alignment zone behaviour at the critical PPMs.

Processing Multiple Glyphs

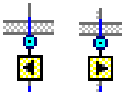
You can convert Type 1 hints to TrueType instructions for many glyphs at once if you use the Transform dialog box.

Open the Transform dialog box with the **Tools/Transform** command and select the **Convert to Instructions** command in the **Hints and Guidelines** section.

Hinting Sidebearings

With FontLab's TrueType hinting tool you can set instructions that will control the horizontal position of a character's sidebearings. This gives you precise control over a character's metrics at small PPM sizes.

In the editing field of the Glyph window you can see yellow marks that designate sidebearing points. There are left and right sidebearing points. The left mark controls the position of the left sidebearing and the right mark controls the right sidebearing.



Sidebearing points are always aligned in the horizontal direction.

You may apply any command to the right sidebearing point, but the left point can be used only as a source of links or interpolate commands.

By moving the right sidebearing point you can control the amount of white space at the right side of the character.

Usually the position of the right sidebearing point is controlled by a rounded single link command or by a final delta instruction. Because the length of the single link with rounded distance cannot be less than one pixel you can set this command and be sure that left or right whitespace will never disappear.

Both sidebearing points may be used as sources of any command just like any other point of a character's outline.

How FontLab Implements Commands that Move Sidebearing Points

The TrueType font format has a special table that can contain the widths of characters that are rendered at specific PPM sizes. This helps the rasterizer to preview lines of text on screen very quickly. FontLab builds this table when it exports a font in TrueType format. To do this it interprets instructions that are applied to the sidebearing points at every PMM size from 9 to 24.

Hinting Composite Glyphs

FontLab 4 allows you to set TrueType hints for the composite glyphs. There is not much visible difference between hinting plain and composite glyphs – Glyph window and tools look as if you are hinting decomposed glyph.

The principle difference is that components of a composite glyph appear already hinted. It means that if you have hinted glyph ‘E’ and hinted glyph ‘caron’ and if you want to check hinting in glyph ‘Ecaron’ you will see that both components look exactly as they look in respective glyphs, and the only thing you need to do is to hint their relative position.

Usually all you need to do to hint a composite glyph is to put a few middle-delta instructions on the contours of accent components so they will get a correct and symmetric position in the composite glyph.

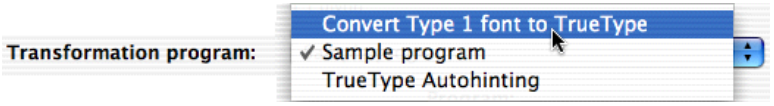
You can however completely override hinting of components and provide new hinting program for a composite glyph. In this case you may take a composite glyph as a combination of all components as if it is decomposed.

Automatic TrueType Hinting

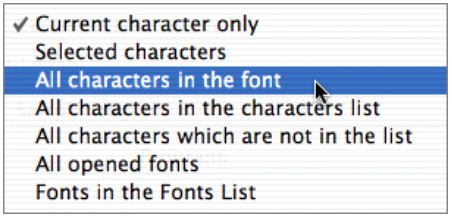
There is no special command “TrueType autohinting” in FontLab. There is Type 1 autohinting and automatic conversion of Type 1 fonts to TrueType instructions. This is a more flexible solution to the TrueType autohinting problem because it allows you to use all the correct Type 1 hints that you have or to build both hints at once.

To automatically generate visual TrueType hints for the whole font:

1. Select the **Transform Range** command from the **Tools** menu.
2. If you have a Type 1 font (outlines of 3rd-order curves) with a Type 1 hints set select **Convert Type 1 font to TrueType** from the program selection popup menu:



3. If the font has no hints at all select **TrueType Autohinting** from the program selection popup menu.
4. Click on the Load button to the right from the program selection popup menu.
5. Choose an application range for the selected program:


✓ Current character only
Selected characters
All characters in the font
All characters in the characters list
All characters which are not in the list
All opened fonts
Fonts in the Fonts List

6. Press the **OK** button to start the process.

Working With Bitmaps

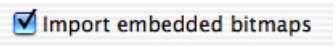
Setting complex instructions program and multiple delta instructions may take a lot of time. An alternative is to create bitmap versions of the glyphs for most important sizes so TrueType rasterizer will use them instead of rendering outline glyphs and instructions program.

Bitmaps can be defined for any combinations of glyphs and PPMs. A font may have only one bitmap for one glyph and one PPM or lots of bitmaps for different glyphs at different PPMs – there are no limits.

FontLab supports import, export and editing of so-called embedded bitmaps.

Importing Bitmaps

Import happens automatically if this option is active in the **Application/Preferences/TrueType** dialog box:



When a font has embedded bitmaps, PPMs for which bitmaps are present are colored red in the TrueType tool Preview panel:

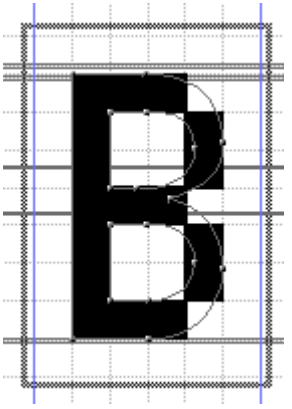
You can import embedded bitmaps from a different TrueType font. To do so:

1. Click on the button on the TrueType hinting toolbar to enter the bitmap mode of the TrueType hinting tool.
2. CTRL-click anywhere in the Glyph window and select the **Import bitmaps...** command in the context menu.
3. Select a TrueType font with embedded bitmaps in the standard File Open dialog box.

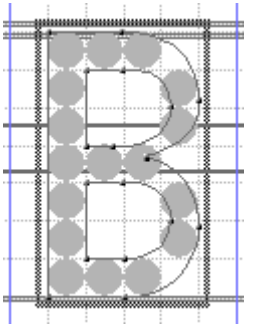
Editing Bitmaps

To get access to embedded bitmaps activate Bitmap mode by clicking on this button on the TrueType hinting toolbar: .

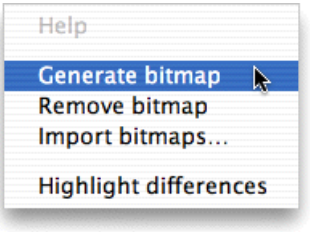
You'll see the Glyph window and preview panel change:



In Bitmap mode the Glyph window previews the bitmap with pixels represented as black squares. If there is no bitmap defined for the current PPM, it looks different:



If you want **to add a bitmap for the current PPM**, Ctrl-click anywhere in the Glyph window and select the **Generate bitmap** command in the context menu:



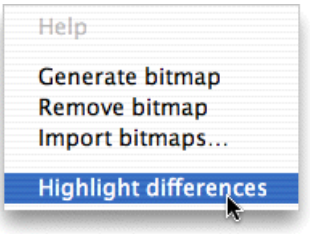
You can edit the bitmap with the mouse: just click and drag the mouse.

To remove the bitmap for the current PPM use the **Remove bitmap** command in the menu.

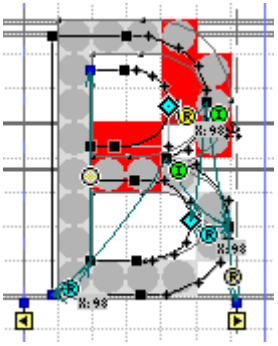
When there are bitmaps defined for the glyph, they are highlighted in the Preview panel:

Highlight Differences

You may highlight the differences between the result of the rendered and hinted outline and a bitmap. Enter the Bitmap mode and select the **Highlight differences** command in the context menu:



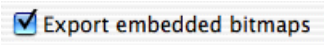
When you leave the bitmap mode (click on the button or select any other TrueType hinting tool) you will see that all pixels that are different in the rendered outline and in the bitmap are highlighted in red:



You can use this feature to separate the “artistic” and technical parts of the hinting process: make good bitmaps; activate the Highlight differences feature; go into hinting mode and set hints so there are no red pixels.

Exporting Bitmaps

To export embedded bitmaps make sure that this option:



is active in the **Application/Preferences/TrueType/TrueType Export** dialog box.

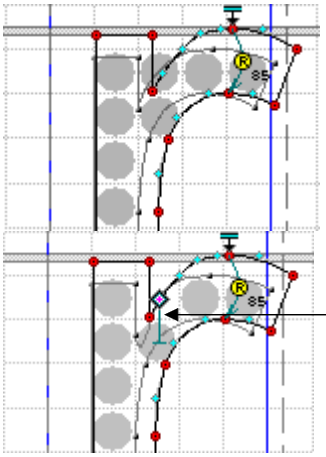
Please note that the operating system will display embedded bitmaps only when a font is rendered in black/white mode. If font smoothing is activated in the system and allowed for the point size in the font, bitmaps will not appear.

Hinting Strategies

This section will give some examples of visual hinting strategies that are applicable in different situations.

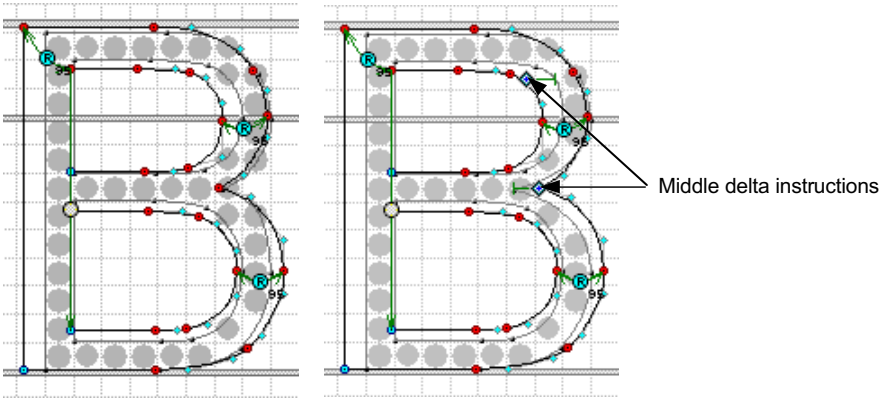
Middle Delta or Final Delta

The final delta command is the very last command of the instruction sequence. This command moves only the point to which it is attached and only on PPMs for which it is set. We recommend using this command in situations where only one pixel needs to be “switched” on or off:



The middle delta instruction is much more flexible. You can use it to slightly modify the interpretation of the hinting program because it is automatically inserted between commands and is interpreted after the command that sets the position of a point and before any command that uses that position as a basis (like link and interpolation instructions).

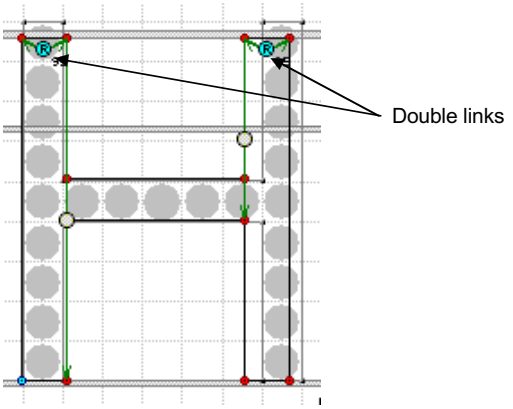
But even if the middle delta command is set on an untouched point it is applied to the point before final interpolation of the untouched points. This point becomes touched after interpretation of the middle delta command so the position of other untouched points will be changed proportionally to the shift of that point:



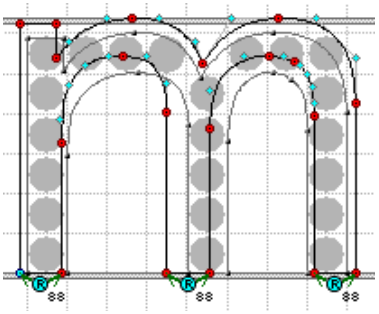
We recommend using the middle delta command with untouched points when it's necessary to shift several points at once. Usually you have such a situation on diagonal and curved stems.

Single Link or Double Link

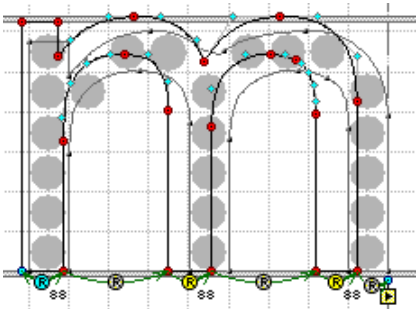
We recommend using the double link command only in situations where only the width of the stem is important, not the position:



Here is a good example of the situation where **double links should be replaced by a chain of single links**:

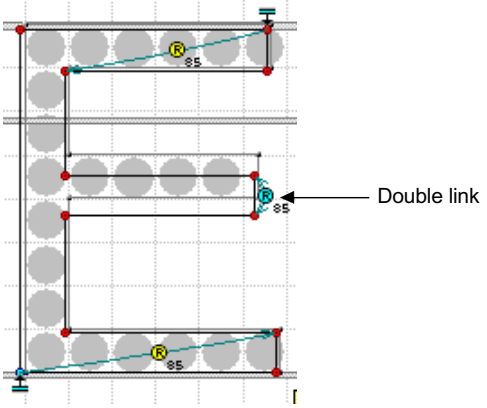


mmmmmmmmmmmmmmmmmmmmmmm



mmmmmmmmmmmmmmmmmmmmm

Another possible application of the double link is hinting a character's middle horizontal stem:

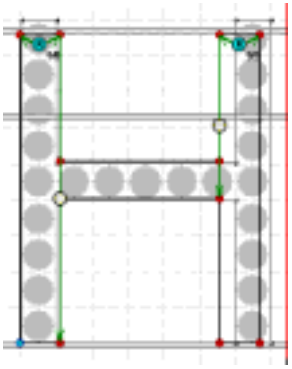


If the position of the middle stem in different characters is different, you cannot be sure that the hinted position will be the same at all PPMs, even if the difference is very small. There is always a chance that at some PPM the rounding of linked points will be different and the link will be shifted top or bottom.

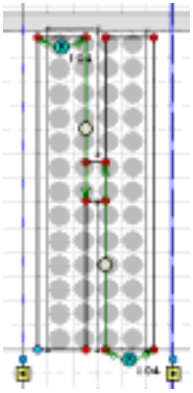
Be ready to set a number of middle delta instructions or, better yet, set a special middle line alignment zone and use a combination of align and single link instructions.

Hinting White Space

Sometimes it's necessary to set the width of the white space inside a character, especially with a narrow character that has more than one vertical stem. Take the character "H" as an example. With normal fonts we usually use two double links to set the widths of vertical stems:

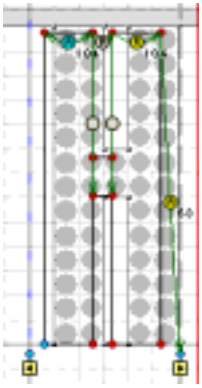


Because at all PPM sizes we will have enough space between stems we don't really care about white space. But look what happens if we try to use the same technique **with a narrow font**:



At many PPMs the vertical stems are snapped together and the character becomes completely unreadable.

We can, however, use only one double link, then link the two stems together with a single link with a rounded distance and set the width of the right stem with a single link connected to the stem:

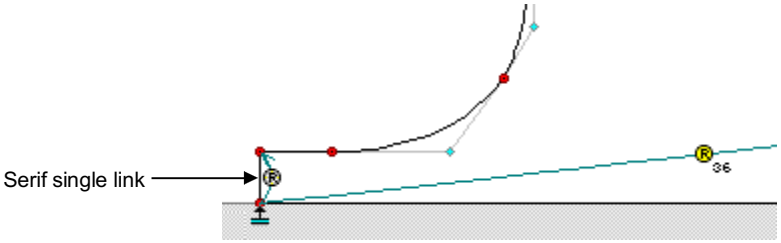


The rounded distance of the single links cannot be less than one pixel, so we will always have some white space between the stems.

To complete the hinting of this character we have to add a rounded single link to the right sidebearing point to make sure that the white space between our character and the next character does not disappear.

Hinting Serifs

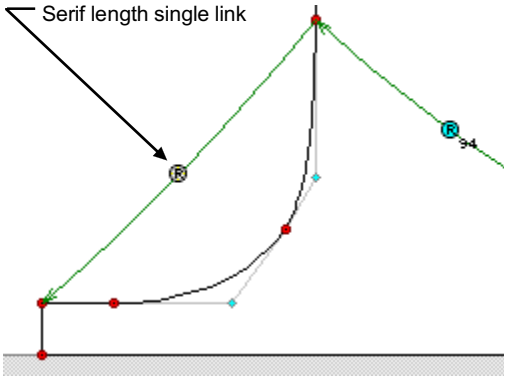
We recommend **hinting serifs in the vertical direction** by a single link command:



If the serifs are thin then it's not necessary to attach their width to the standard stem width. A single link with rounded distance will be enough. Rounded single links cannot appear less than one pixel in length so the serif will never disappear.

Of course, if you want to have more control over the serif's width you can define a special stem width and attach the serif's single links to this width.

For total control over serifs' behavior you can also set horizontal commands that will control the "length" of the serifs:



Hinting Diagonals

Hinting diagonal stems is the most complicated job in hinting. There are no special “diagonal” commands so you have to set both horizontal and vertical commands to control the width of a diagonal stem.

We know two ways to hint diagonal stems:

1. Define special standard stem widths. Set vertical and horizontal commands attached to these widths and control the rounding of stem widths using the stems dialog box.
2. Set the positions of one side of the stem using the usual commands (align and link) and set the position of the other side by single links that are not rounded and not aligned. Adjust the width of the stem using middle delta instructions.

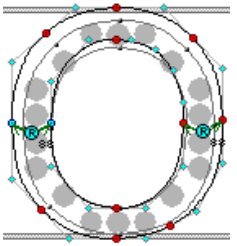
   If a diagonal stem is intersected by another vertical, horizontal or diagonal stem then we recommend using the interpolation instruction to set the position of the intersection points.

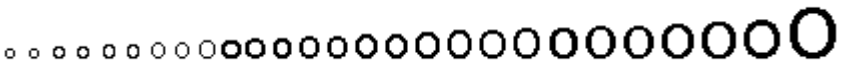
In most cases good hinting of diagonal stems is not possible without final delta-hinting.

Hinting Symmetrical Characters

There are two types of symmetrical characters: round characters (like ‘O’) and “pike” characters (like ‘V’).

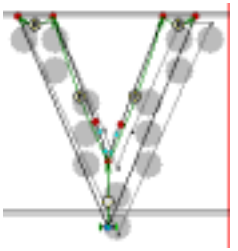
Round characters are “automatically symmetrical” in both directions. So it’s enough to set the positions and width of the vertical and horizontal “stems” and the center points will be interpolated and automatically set in the central position:

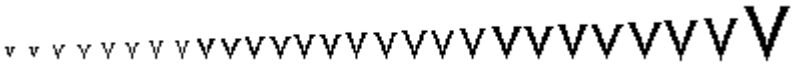




For “pike” characters this is not good enough because the position of the pike point is important. At most PPMs you must set the width of the pike to one pixel. Automatic interpolation of the outline may set the pike point to a position where the pike will disappear.

So in the case of pike characters we have to begin by positioning the pike point with the **align to pixel center** command and then set the positions of all other points by linking to this point:

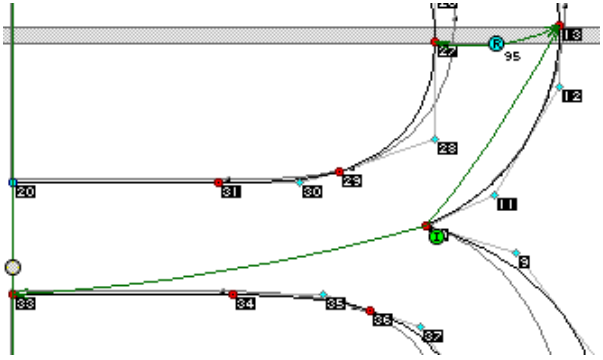




Interpolate or Not

As you already know all points that were not positioned by commands (untouched points) are interpolated automatically by the final interpolation command. Final interpolation sets the position of any untouched point proportionally to the closest untouched points. But sometimes you may want to control the interpolation process and directly set the base points of the interpolation. Usually this happens when a point is geometrically between two touched points that belong to different contours:

The only way to keep point 10 between points 31 and 34 is to set an interpolation command in the vertical direction. It's a good idea to interpolate the same point in the horizontal direction too:



Multiple Master Fonts

With FontLab you can make a special type of font called a Multiple Master font. Multiple Master fonts contain several font styles, called *master fonts*, in one font file. A program that uses a Multiple Master font can select not only one of the master fonts, but also any intermediate style created by interpolation of the master fonts. So it can use not only Bold, Normal, Narrow or Wide styles, but any style in between, such as 30% Weight and 47% Width. We will discuss this process in full detail in the next sections.

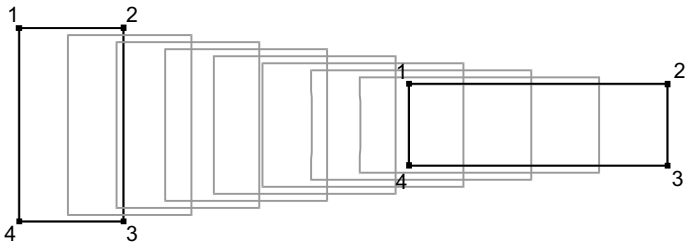
The Multiple Master font format is an extension of the Type 1 font format. This means that Multiple Master fonts (or MM fonts, for short) are compatible with all PostScript printers. MM fonts can be used on Windows or Macintosh computers with the help of Adobe Type Manager (ATM) versions 3.0 or later. (Unfortunately, according to Technical Note TN2024 Mac OS X doesn't yet support MM fonts.)

For technical details and an excellent background in this subject we strongly recommend you read "Designing Multiple Master Typefaces", a book published electronically by Adobe Systems.

It is available at:
http://partners.adobe.com/asn/developer/pdfs/tn/5091.Design_MM_Fonts.pdf

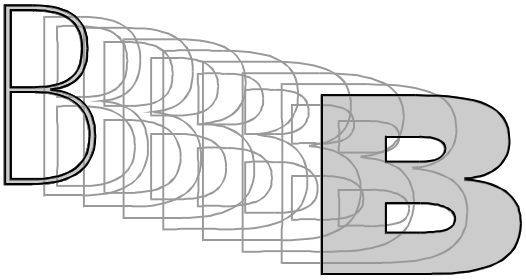
Multiple Master Fonts Theory

A simple picture can describe the main idea of the Multiple Master font format:



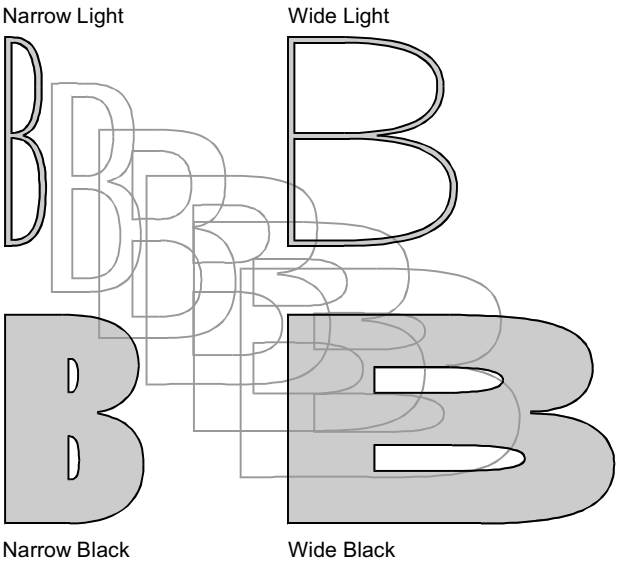
If we have two contours that have the same number of points of the same type (vector or curve), we can always make intermediate contours by interpolation. The number of these different contours is limited only by the precision that we choose.

Applied to fonts, this idea gives us the following:



In this picture we used two styles of the same typeface, Light and Black, as ends and, as you can see, we have many more styles in the middle. Fonts that are used to define end designs are called *master fonts*. A combination of several master fonts into one font file is called a *Multiple Master font*.

In the example we used only two master fonts, and we changed only one property of the font using interpolation. But we can define more master fonts and get even more choices for changing the design of the font. For instance here's what happens when we use four master fonts: now we can vary both the fonts' width and weight:



The quick brown fox jumped over the lazy dog
The quick brown fox jumped over the lazy dog
The quick brown fox jumped over the lazy dog
The quick brown fox jumped over the lazy dog
The quick brown fox jumped over the lazy dog
The quick brown fox jumped over the lazy dog

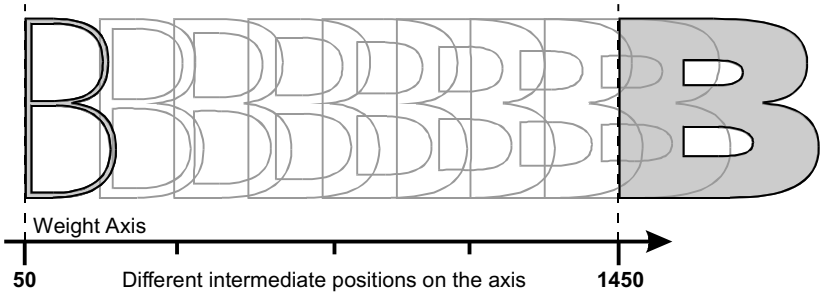
The text sample typed with different instances of the same Multiple Master font

Design Axes and Dynamic Range

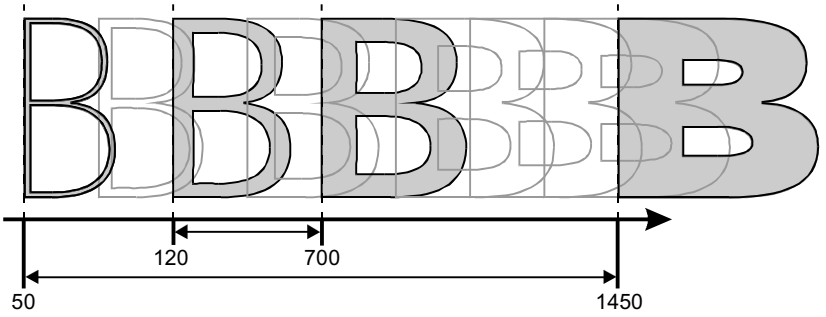
It is possible to define a number of properties of a MM font using master fonts. These properties are called *design axes*, or simply *axes*, for short.

You can define as many of these design axes as you want, but four axes in a single MM font is the practical limit since there is no program as of this printing that can use more than that. The number of masters that is necessary to define a given number of axes can be calculated as 2^n, where n is the number of axes. So a single axis font requires 2 masters, a two-axis font – 4 masters, a three-axis font – 8 masters and finally, a four-axis font requires 16 masters.

You can see in this picture how design axes and master fonts are used to select an intermediate design of the MM font:



In MM fonts it is important to know how much the property connected with the design axis can be changed. If we take the Weight axis, for instance, then it might be changed from normal (120) to bold (700), a moderate change, or from light (50) to black (1450), a much larger change:



The range within which the property of the axis may be changed is called the *dynamic range*. It is the distance between the two masters that define an axis measured using a standard scale line.

In our example the first dynamic range is 580, from 120 to 700, and the second dynamic range is 1400, from 50 to 1450.

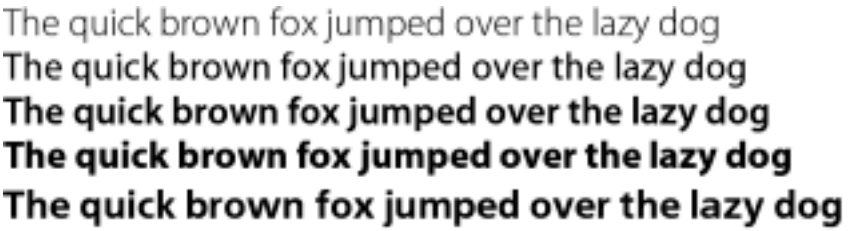
So that desktop publishing programs can use MM fonts that were designed by different manufactures some standards have been set for axes types and names.

Standard Axes

There are four standard axes defined by the Multiple Master standard. They are supported by Adobe and thus are likely to be compatible with all current and future desktop publishing programs. Of course, you can define your own design axes, but you must register new axis types with Adobe.

Weight

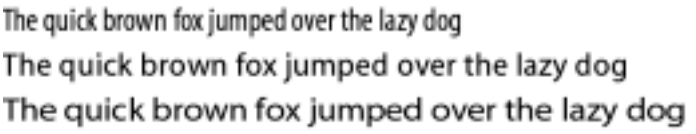
This axis allows you to change the weight of the font:


The quick brown fox jumped over the lazy dog
The quick brown fox jumped over the lazy dog
The quick brown fox jumped over the lazy dog
The quick brown fox jumped over the lazy dog
The quick brown fox jumped over the lazy dog

Changes in the weight (the defined dynamic range) may be very different in different fonts, for example, from normal to bold or from light to black or from bold to heavy, depending on the designer's intentions.

Width

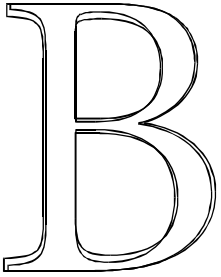
This design axis allows you to change the width of the font:


The quick brown fox jumped over the lazy dog
The quick brown fox jumped over the lazy dog
The quick brown fox jumped over the lazy dog

This is very useful in making text fit in a given space on the page. If the Condensed style is too narrow and the Normal style is too wide, with a MM font you can select an intermediate width style and the text will fit perfectly.

Optical Size

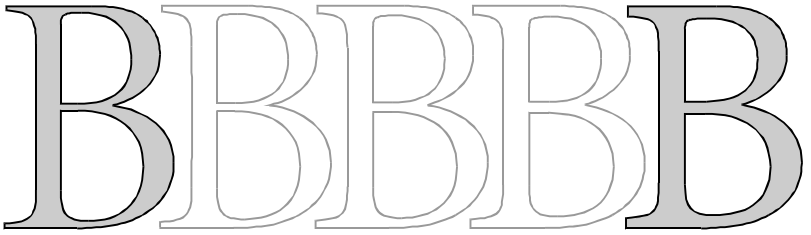
This is an axis for high-end typography. In the pre-computer age typographers used metal fonts that were aligned in strings manually or with the help of complex mechanical machines. In those times fonts were not "scalable" and there was a separate physical typefont for each point size of a typeface. And fonts that were designed for different point sizes of the same typeface had slightly different designs for improved legibility:



Designs of the same character for 6 and 72 pt. point size

When scalable fonts appeared this typography feature wasn't compatible with the technology so it all but disappeared. We had scalable fonts that could be shrunk or enlarged to any point size, but the price for this was a little decrease in the quality of typeset text.

With Multiple Master fonts it is possible to define two optical size master fonts, one for small point sizes and another for large point sizes, and use MM interpolation to create the proper font design for the final selected point size:



72 pt. master font 6 pt. master font

In the following example you can see two lines of text typed with the same font but different optical size selection:

ABCabc The text font
ABCabc The header font

For better comparison, both lines were typed at the same point size (28 pt) but with the different optical sizes. Note the different contrast and inter-character spacing.

Style

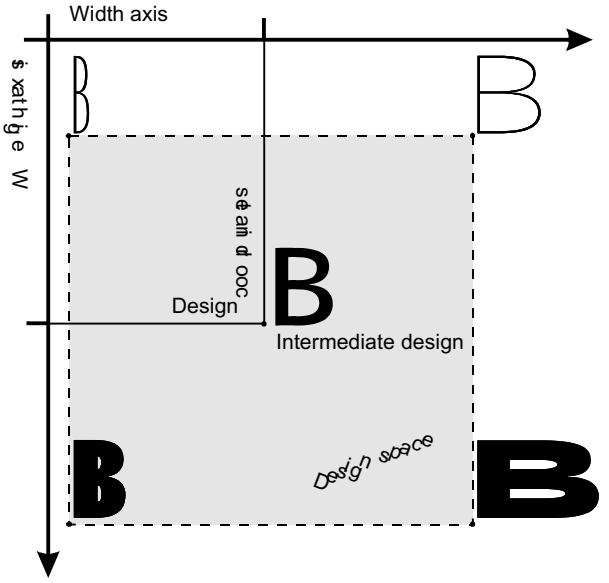
This design axis can cover many different design properties. Here's one example of what you can do with it:

There is no standard definition or real property of this axis. And there is no standard for its dynamic range.

Design Coordinates and Weight Vectors

You know that all axes have ranges. These ranges can vary from 0 to 9999, but a more practical range of 0 to 1000 is usually used. Somehow we have to define a way to designate the position of an intermediate MM design in the design axis space.

A very natural way to define the position of the intermediate font is to use its position on each design axis. These coordinates are called *design coordinates*:



There is also another type of coordinate called *blend coordinates*. Blend coordinates are normalized to the given dynamic range of the axis. If an axis has a (50-400) dynamic range then the design coordinate (50) equals the blend coordinate 0.0. The design coordinate (400) equals the 1.0 blend coordinate. Intermediate design coordinates are converted to the blend coordinates using linear calculation (by default) or a non-linear axis graph (described in the next section).

When we know the design coordinates we can calculate what proportion of each master font we need to build the intermediate font. This process is called blending.

Simple formulas are used to make these calculations and as a result we get an array of values that define the "weight" of each master in the final interpolation that produces an intermediate font:

[0.13, 0.77] – for a one-axis font (two masters are blended)
[0.02, 0.24, 0.11, 0.63] – for a two-axis font (four masters are blended)

Note that the sum of all weights is always equal to 1.

This array of masters' weights is called the *weight vector* and is used internally by the font interpreter to build the intermediate font. Note that while blend coordinates are used internally, externally, design coordinates are used.

A special weight vector, called the default weight vector, should be set in the font header to define a standard intermediate design of the MM font. A MM font is "seen" as a typical Type 1 font by a PostScript device or program, and this Type 1 font is an intermediate instance of the MM font with the default weight vector.

Extrapolation

The Multiple Master font specification allows only *interpolation* of the master designs to define an intermediate font. In addition to that basic feature, however, the multiple master font structure allows linear *extrapolation* of the designs also. This feature allows us to artificially extend the design range of the MM font, which sometimes produces interesting results:

Standard dynamic range for the Width axis

Extrapolated range for the Width axis

Extrapolated Weight

In real life this feature lets you extend font families with very little effort: for instance, build an MM font using the normal and bold styles and then add light and extra bold using the extrapolation feature.

Anisotropic Interpolation

Standard MM fonts have the same interpolation values for both the X and Y directions. In FontLab it is possible to define a different interpolation weight for directions. With this you can make interesting form variations:

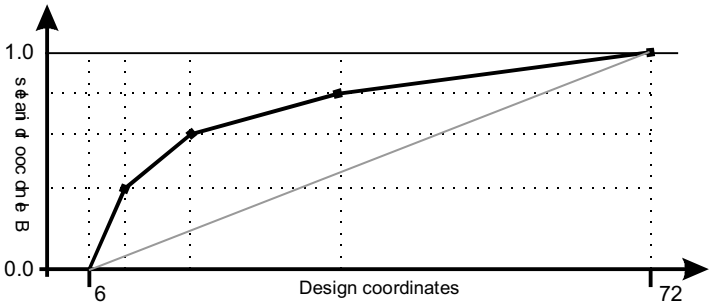
Standard (proportional) interpolation

Anisotropic interpolation

As with the extrapolation, this feature is not a part of the MM specification and anisotropic interpolation (or extrapolation) will work only inside FontLab. You can use it to generate single-master fonts from MM families.

The Axis Graph

Usually design coordinates are translated to the weight vector using simple linear calculations. It is possible, however, to define a map for non-linear calculations:



This map sets a series of points that map the design and blend coordinates. Between these points linear interpolation is used (this is called a piece-wise linear function).

Usually this kind of *axis graph* (or *axis map*, in some literature) is used for the Optical Size axis. The Optical Size axis usually has a dynamic range of 6-72. The design coordinate 6 means the version of the typeface adjusted for 6 pt. type. The design coordinate 72 represents the typeface design for 72 pt. type. Between these values the blend coordinates are mapped using the axis graph, as in the picture above. This mapping gives better control of the variation of the design for the different design coordinates. In this case it causes a more rapid change in the appearance of the typeface at smaller optical sizes than at larger.

The value of the leftmost and rightmost design coordinates of the axis graph (6 and 72 in our example) are always mapped to 0 and 1 blend coordinates and are used by the font interpreter to get information about the dynamic range of the axis. So an axis graph is always present for each axis but in most cases it is just a single straight line.

We again recommend you read the “Designing Multiple Master Typefaces” book to get more information about the axis map.

Multiple Master Fonts in FontLab

As we said before, in FontLab you can make Multiple Master fonts that will be completely compatible with the Multiple Master specification. You can edit every aspect of the MM fonts, from master outlines to metrics to kerning and font header data. You can make up to four axes, which means that the font will need 16 master fonts. And at any time you can view an intermediate font made from the designed MM font in its current state.

In the following sections we will discuss MM-specific FontLab features. We presume that you have read the previous chapters of this book and know all the single-master features of FontLab.

Creation of MM Fonts in FontLab

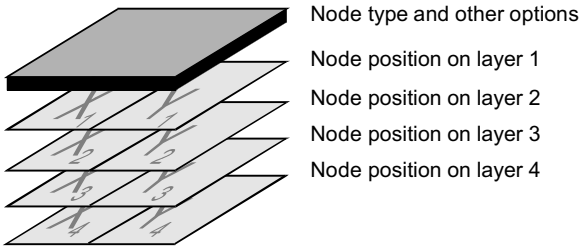
In FontLab you can do the following MM-specific things:

1. Open any existing MM font for editing.
2. Convert a single-master font to a MM font.
3. Define additional axis in a MM font.
4. Remove any of the axes of a MM font.
5. Convert a MM font to a single-master font.

You can also use FontLab's special feature, called **Mask to Master,** to simplify the combination of several single-master fonts into one MM font.

Multiple Master Outlines in FontLab

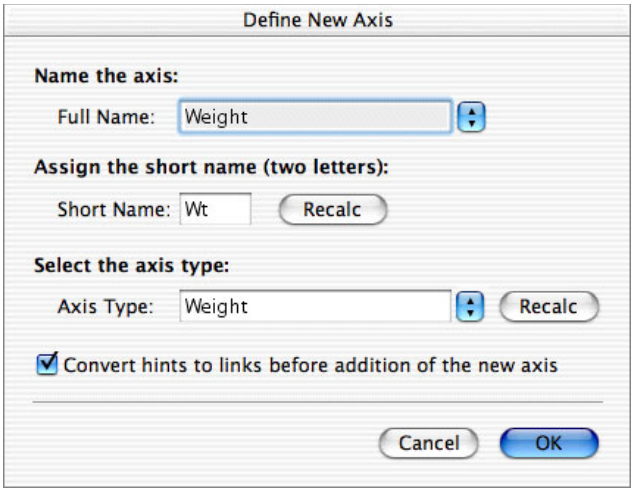
In FontLab all outlines are multiple master-compatible from the very beginning. Every node of the outline has a layered structure:


Node type and other options
Node position on layer 1
Node position on layer 2
Node position on layer 3
Node position on layer 4

Thus, a Multiple Master font in FontLab is not a combination of several separate master fonts, but a single multilayered font. This means that in FontLab it's impossible to make master fonts incompatible, because they are just the different layers of the same font.

Defining an Axis

If you want to convert a single-master font to a Multiple Master font you must define a new design axis. Select the **Tools/Multiple Master/Define New Axis** command. You will see a dialog box:



To define an axis you must enter an axis name, a short name and select an axis type.

Select the name of the axis in the **Full Name** popup menu. You can enter your own custom name, but we strongly recommend you use only names that are in the list in order to make the font compatible with font interpreters. When you are adding a new axis to a Multiple Master font the names of the axes that are already present in the font will *not* appear in the popup menu.

The short name of the axis is used when a font interpreter needs to make a name for an intermediate font. Usually the name looks something like:

MinionMM_245 wt 580 wd

where **MinionMM** is the name of the MM font, **245** and **580** are design coordinates of the intermediate font and **wt** and **wd** are short names of two axes, Weight and Width respectively.

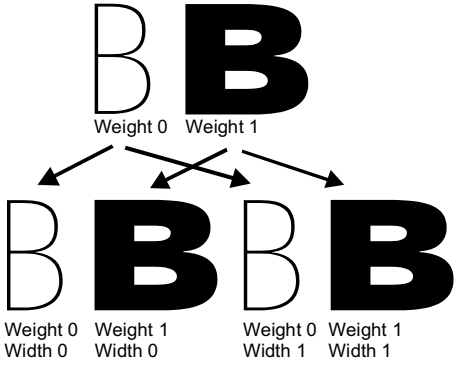
The axis type is necessary to identify a new axis. As we said in previous sections, there are four registered axis types:

Weight	Variation of the font's weight
Width	Variation of the font's width
Optical Size	Variation of the font's design to make it more legible at different font sizes
Serif	Modification of the font's style. Usually this is used for modification of the font's serifs. For example, from serif to sans-serif font, or variation of the serifs' width.

Usually it is not necessary to fill in all three edit fields. Just select or enter the axis name and FontLab will automatically fill in all the other fields.

Check the **Convert hints to links before adding new axis** option to automatically apply the **Convert Hints to Links** action to all characters of the font. We recommend converting hints to links because in a MM font it is much easier to edit links than hints. Links are Multiple Master-compatible by default, because they connect multilayered nodes, while hints connect points that have different coordinates on different masters.

When you define an axis the number of masters will be increased twofold. Therefore a single-axis font will have two masters, a two-axis font will have four masters, etc. The contents of all existing masters will be duplicated, as seen in the picture:

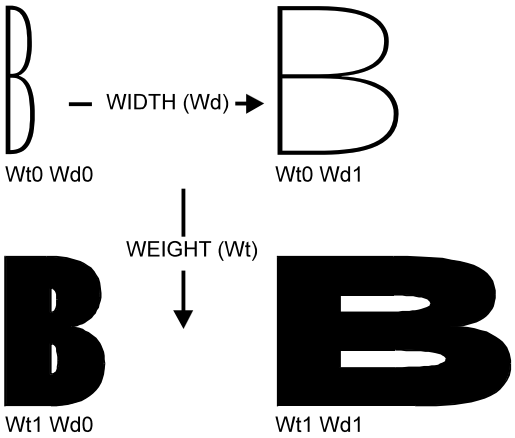


Selecting a Master Font

When you define an axis or open a Multiple Master font you will have more than one master font. Open a Glyph window with any character and you will see several masters appear in the editing field and the master selection popup menu enabled.

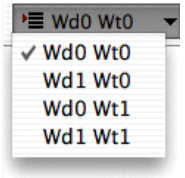
Different masters of the multiple master font appear as outlines of different colors. One master appears in black and it is the only editable master, called the active master. All editing tools, actions and operations are applied to this master.

All masters are identified by their names. The master name is the position of the master on the design axis:

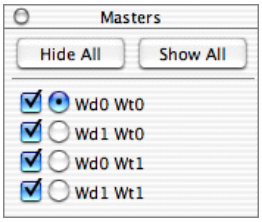


You can select the active master using the following methods:

1. Double click the master outline that you want to activate or
2. Select a master in the master selection popup menu in the Glyph window header:



3. Select the **Window/Masters Panel** command. The Masters panel appears:



Click on a master's radio button in the Masters panel.

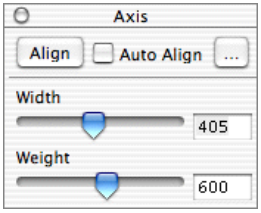
Using the Masters panel you can also select which masters will appear in the Glyph window. Use the masters' check boxes in the Masters panel to show or hide masters.

Press the **Hide All** button in the Masters panel to hide all masters except the active master or press the **Show All** button to show all the masters.

Using an Axis Panel

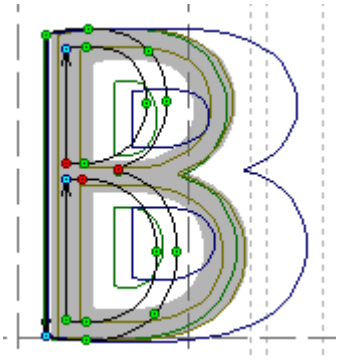
When the Preview mode of the Glyph window is active you can display an intermediate instance of the MM font that is selected for editing.

Open the Axis Panel by using the **Window/Axis Panel** command. You will see the Axis panel appear:



This panel has a slider and an edit box for each axis in the MM font. Use these controls to select the design coordinates of the intermediate instance that you want to preview.

The filled preview of the selected intermediate design will appear in the Glyph Window in gray:



Four masters' outlines and an intermediate preview in gray

If an active master is the same as the selected instance (as when master Wt0 Wd0 is active and both sliders are in the leftmost positions), the filled preview will appear in black because it precisely corresponds to the active master.

To preview the active master (in black) press the **Align** button in the Axis panel. Switch on the **Auto Align** check box to make alignment of the intermediate preview to the active master automatic, so that when you select another master as active, the preview will automatically follow it and you will see a filled preview of the master that you are working on.

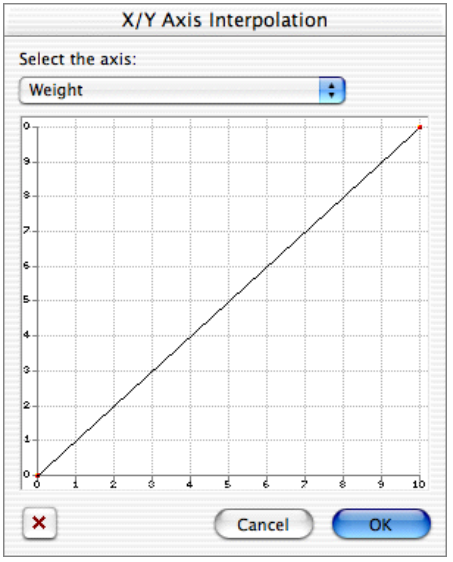
Extrapolation

The Axis panel allows you to select not only design coordinates that are within the axis dynamic range, but to go beyond that point, thus extrapolating the master designs.

If one of the values is out of defined dynamic range, the editing control background turns yellow:

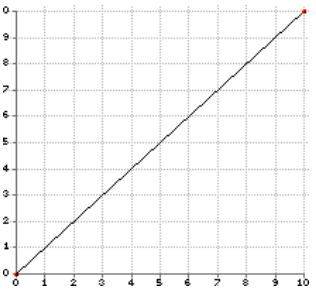
Anisotropic Interpolation

Click on the ... to open the anisotropic interpolation dialog box:

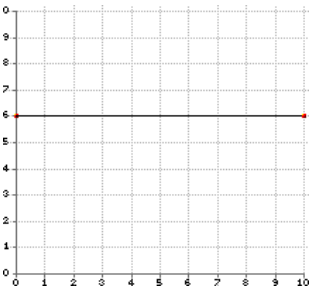


Select the axis in the list at the top and use the graph to define the relationship between X and Y interpolation.

The X-axis on the graph represents the X interpolation position on the design axis. The Y-axis represents the relative Y interpolation. Here are some examples of the possible X-Y interpolation relationships:



Isotropic, proportional interpolation. This is the default setting.



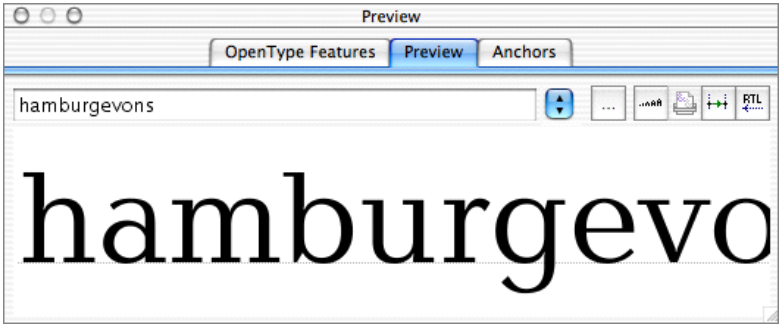
Fixed Y (width) axis. Width does not change when you move the slider. The weight (X axis) changes linearly along the weight axis.

To edit the graph, use same mouse commands as when you are editing a glyph's outline. **CMD+OPTION**-click inserts a point. **CTRL**-click when a point is being moved removes the point.

Click on the button to reset any changes and return the graph to the linear state.

Previewing the Intermediate Design

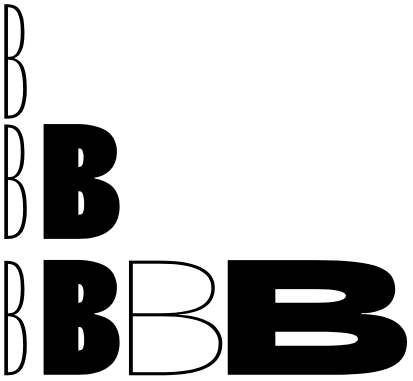
You can use the Preview panel to preview an intermediate design of a MM font in high-quality mode. Use the **Window/Preview panel** command to open the panel:



Please, refer to page 269 in the “Glyph Window” chapter for more information about the Preview panel.

Designing Master Fonts

In FontLab you can make a Multiple Master font by “raising” it from a single master font by adding an axis. When you have a one-axis MM font you can add another axis and so on:

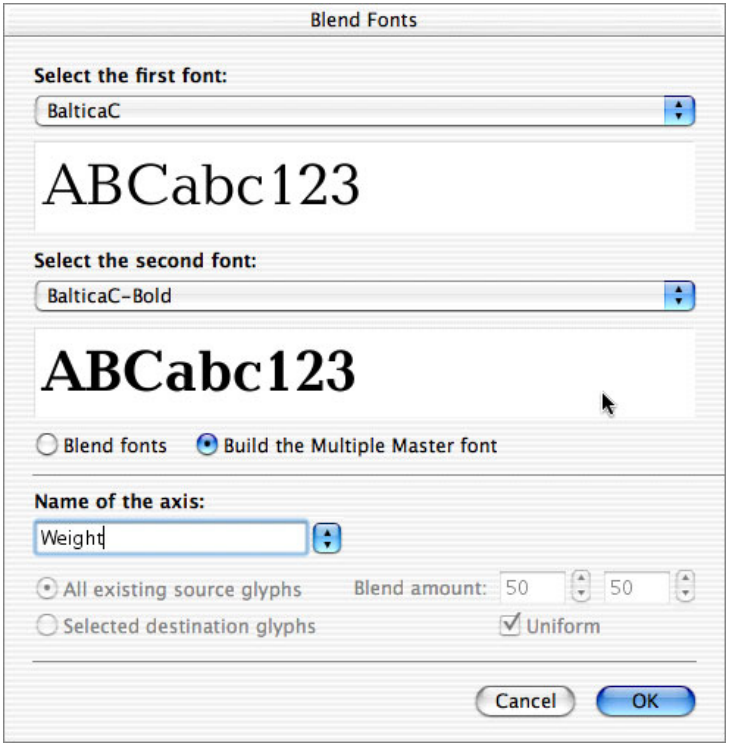


There is no automatic way in FontLab to take two fonts and make a MM font from them using one as the first master and another as the second master. This approach has pros and cons. On one hand, you always get a properly made MM font. There is no chance that one of the font’s characters has not been converted according to MM rules. On the other hand, this process takes a lot of time.

We have, however, included a special feature in FontLab that can help you make MM fonts faster from different fonts.

Using the Blend Feature to Build MM Fonts

If you have two single-master fonts and want to build a Multiple Master font you can use the **Tools/Blend Fonts...** operation. Open the fonts and select the **Blend Fonts...** command. You will see the Blend Fonts dialog box:



Select the first and second fonts in the popup menus in the top area of the dialog box and choose the **Build the Multiple Master font** option.

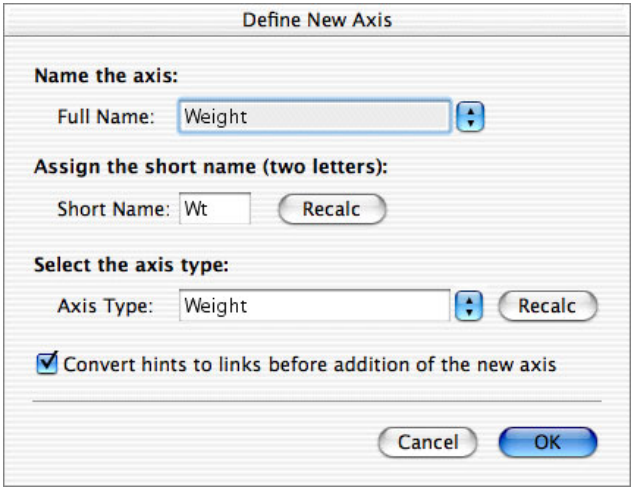
Enter the name of the Multiple Master font axis in the **Name of the axis** control. You can select one of the standard names or you can enter any other name.

Click the **OK** button and wait while FontLab generates a font. Some glyphs cannot be blended (usually these glyphs have different number of contours). If they are present, FontLab will show a warning dialog box and these glyphs will have the first font on the outline layer and second font on the Mask layer, so you can correct outlines and use the **Mask to Master** feature (described below) to build the MM font.

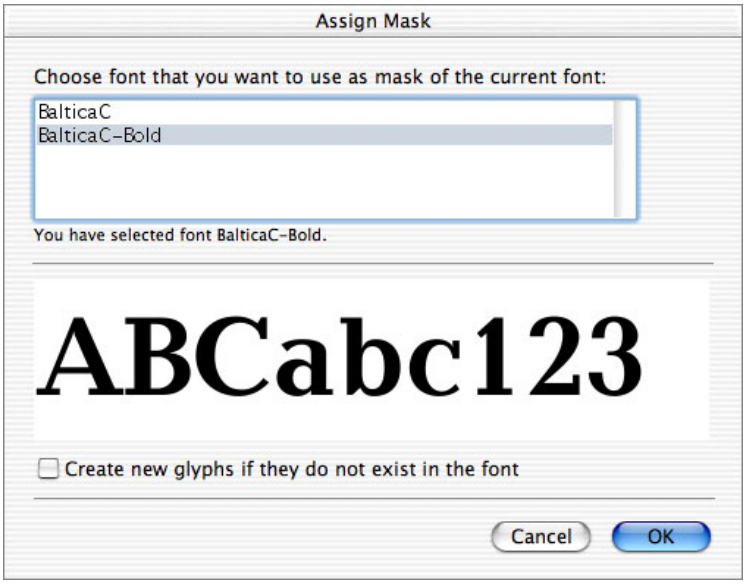
Mask to Master Action

You can use a mask layer to construct a MM font from the existing styles of a typeface. Let's take a typeface that has Normal and Bold styles and show how to make a MM font with the Weight axis:

1. Open the Normal style of the typeface.
2. Open the Bold style. Now you have two fonts open.
3. Activate the Font window that contains the Normal typeface and define a new axis: Weight



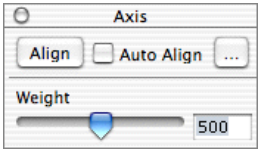
4. Open the Assign Mask dialog box (select the **Tools/Assign Mask** command) and assign a mask from the Bold style:


Assign Mask
Choose font that you want to use as mask of the current font:
BalticaC
BalticaC-Bold
You have selected font BalticaC-Bold.
ABCabc123
Create new glyphs if they do not exist in the font
Cancel
OK

5. Close the Bold style font.

Now you have a Multiple Master font that has one axis, two masters (that are still the same) and a Bold style on the Mask layer. Don't forget to switch on the Mask layer in the Show Layers toolbar.

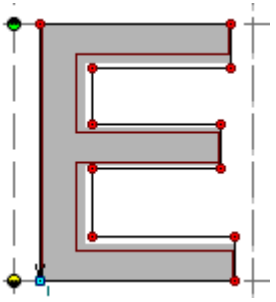
Open any character for editing and activate the Wt1 layer, switch on preview mode and select an intermediate position of the design coordinates in the Axis panel:



Look in the top-right area of the Glyph window. There is a special mark (rightmost) that may be gray, red or green:

A red mark shows that the mask layer is not compatible with the master. “Incompatible” means that mask and outline layers have different numbers of contours, so FontLab cannot match the points and curves.

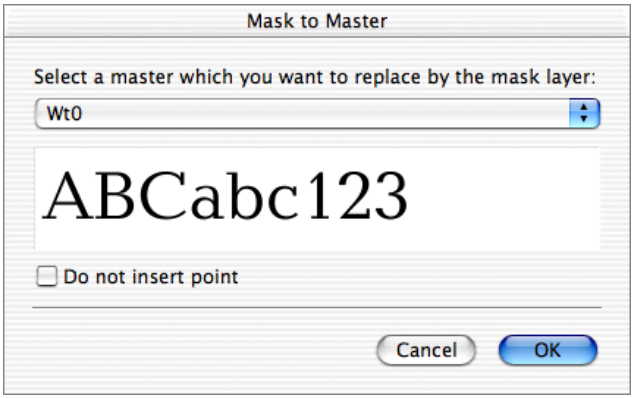
A green mark means that the mask and outline layers seem to be compatible and you may apply the **Mask to Master** action. This action replaces the contents of the active master with the contents of the Mask layer. If the Mask layer is completely compatible with the outline you will instantly get a properly made master for the active layer:



FontLab uses the Blend algorithm to match points in the Mask and outline layers. Sometimes it can produce incorrect results, so we recommend you check your outlines after you complete the Mask to Master operation.

You can apply the Mask to Master action to several characters at once:

1. Activate the Font window. Note the mask compatibility marks in the top-right corner of the characters' cells.
2. Select the characters to which you want to apply the Mask to Master action.
3. Click on the **Tools/Templates/Mask to Master** command.
4. A dialog box appears:



Select the master that you want to replace by the Mask layer in the masters list box. Press the **OK** button to begin the transformation.

Note that when the **Mask to Master** command is used, the value of the width of the character in the Mask layer is copied to the active master. To copy *only* the width of the Mask character (this is always possible regardless of mask/master compatibility), press the **Cmd+Shift+J** key combination on the keyboard.

Using Interpolation to Make Masters

If two styles are completely incompatible you can use the Interpolate operation (described in the “Glyph Window” chapter) to make master fonts.

Put one style on the outline layer and define an axis. Put the other style on the Mask layer using the Assign Mask dialog box.

Select the master that should look close to the appearance of the style on the Mask layer and activate an Interpolation operation. Use Interpolation to “stretch” the outline on the Mask layer. This is relatively easy to do because the destination points of the interpolation links will stick to the Mask layer.

After you complete work with the Interpolation tool use the Edit tool to fine-tune the master.

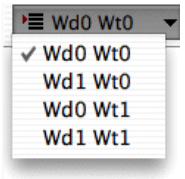
Multiple Master Metrics

You can set metrics for Multiple Master fonts as easily as you can for a single-master font. Of course, you can use the Edit tool to adjust sidebearings but it is much easier to do it with the Metrics window.

When a MM font is opened in the Metrics window its three modes (preview, metrics and kerning) work slightly differently.

The preview mode shows the intermediate font in the sample window. Use the Axis panel to select the design coordinates of this intermediate font.

In metrics and kerning modes only a master font can be previewed in the Metrics window. You can use the Masters panel to select the master to preview and edit or you can select it from the master selection popup menu that looks and works exactly as in the Glyph window:



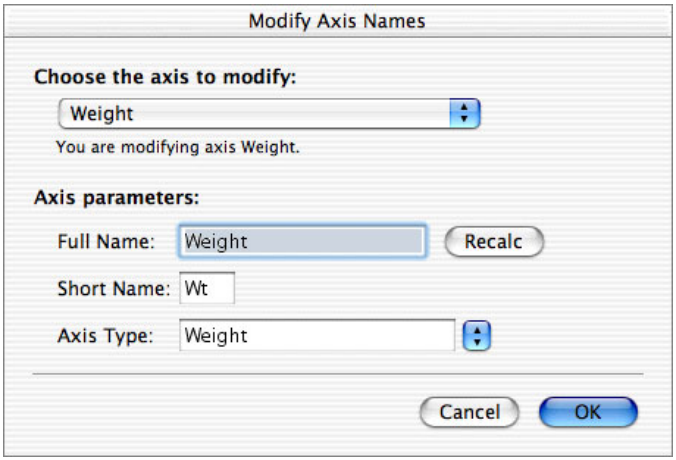
So, to edit metrics and kerning information, select the master that you want and edit it as a single-master font. FontLab will automatically track all changes and complete the Multiple Master data for the font.

Another difference is the save format for the metrics file. If a MM font is open then a binary MMM (Multiple Master Metrics) file is used to store the metrics data. Or you can save a set of text AFM files for each master and linked MM file. Select the format that you want to use in the Format popup menu of the Save File dialog box that appears when you press the **Save** button in the Metrics window.

When you are opening or saving single-master metrics files, they will be applied to (or generated from) the currently active master.

Editing Axis Settings

When you are editing a Multiple Master font you can change the axis names at any time. Select the **Tools/Multiple Master/Modify Axis Names** command and you will see a dialog box:

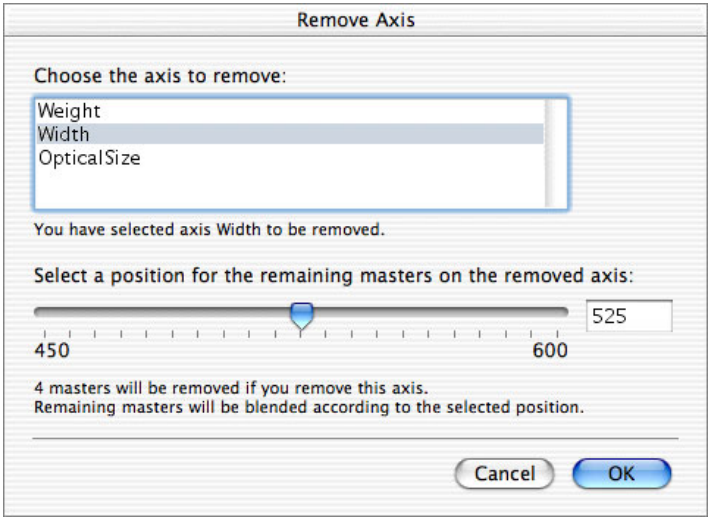


Select the axis that you want to change in the popup menu located in the top area of the dialog box and use the editing fields to change the axis' name, type or short name. Press the **Recalc** button after you change the axis' name to let FontLab automatically calculate the rest of the data.

Removing an Axis

At some point you may want to remove one of the design axes of a Multiple Master font. When you remove an axis the number of masters will be decreased by half. You can lose half of the information also, so be careful with this operation. The remaining masters will be blended according to their position on the removed axis (which you can specify).

To remove an axis select the **Tools/Multiple Master/Remove Axis** command. A dialog box appears:



Select the axis you want to remove in the axis list at the top of the dialog and use the slider to set the “blending coordinate” that will be used to blend the remaining masters.

You can use the Generate Instance feature (described on the page 623) if you want to convert the MM font to single-master font. Sometimes it is easier to use than the Remove Axis feature.

Multiple Master and Font Info

Here is a table of all the pages of the Font Info dialog box an explanation of specific information in a Multiple Master font:

Names

Have minor changes:

1. Multiple Master fonts usually have the “MM” suffix at the end of their names.
2. The length of the Menu name is limited to 7 characters. Use the **Check** button to automatically control the length of the Menu name.
3. MM fonts with a Weight axis have the “All” value in the **Weight** field. MM fonts with a Width axis have the “All” value in the **Width** field.

Copyright

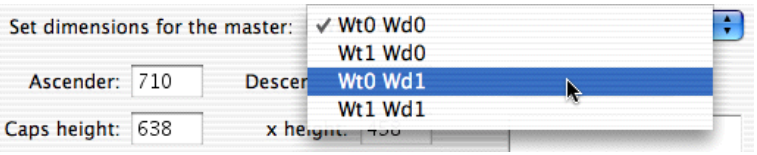
No changes.

Version

No changes.

Dimensions

Each master has its own set of Dimension values. Use the master selection popup menu at the top of the page to choose the Master for which you want to set dimensions:



Alignment Zones and Stems

Each master has its own set of alignment zones, standard stems and other font-level hinting information. Use the master selection list to choose the master whose alignment setting you want to change. Note that when you add new alignment zones they will appear in all masters, so always check all the masters to be sure that the alignment information is set properly.

Panose Identification

Enter settings in the Panose fields Weight or Width if your font has Weight or Width axes respectively.

Codepages

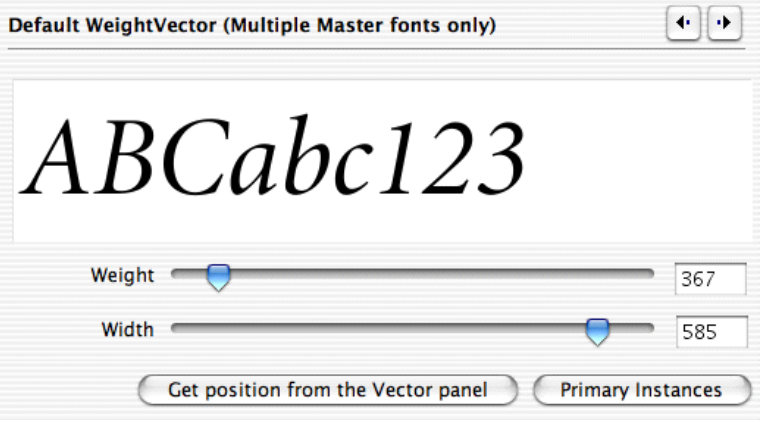
No changes.

TrueType

TrueType fonts cannot be Multiple Master, so no changes on this page.

WeightVector

This page appears only if you are editing the Font Info of a MM font. Use the sliders below the preview window to choose the default Weight Vector of the font.

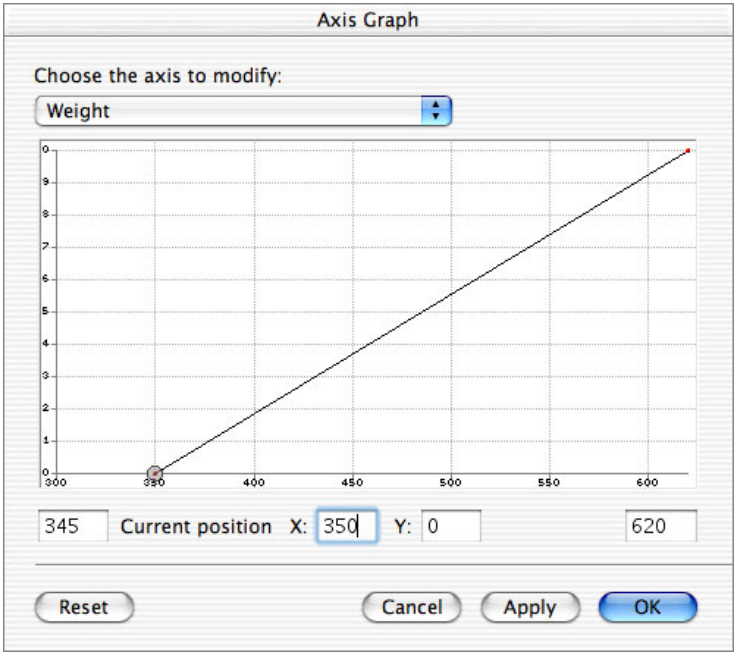


Press the **Primary Instances** button to define more default instances for the font.

Editing the Axis Graph

You can edit an axis graph for every axis defined in a MM font. As we said before, the axis graph is used to set the dynamic range of the axis, so it is always necessary to adjust it.

To edit an axis graph, select the **Tools/Multiple Master/Edit Axis Graph** command. The Axis Graph dialog box appears:



To edit a graph:

1. Select an axis whose graph you want to edit in the axis popup menu located at the top area of the dialog box.
2. Position the mouse cursor on the leftmost point of the line in the graph and click the mouse.
3. In the **Current position** editing fields enter the lowest design coordinate for this axis or position the cursor on the leftmost point and drag it.
4. Do the same with the rightmost point to define the dynamic range.
5. If you want to define a more complex graph you can add points. To add a point, position the mouse cursor on the line segment where you want to add the point and **Cmd-Option**-click the mouse.
6. To move a point, position the mouse cursor on the point, press the button and drag the point to a new position.
7. To enter the position of the point, click on the point and enter the position of the point in the **Current position** editing fields.
8. Press the **Apply** button to accept any changes that you have made.

Choosing Dynamic Range

When you define a design axis you should choose the proper edge design coordinates and dynamic range for it. It is important to make your font compatible with Multiple Master fonts made by other manufacturers. Dynamic range is not so important for the Style axis because there is no standard for it and it is relatively easy to set the dynamic range for Optical Size because it is usually known for this axis. So we have the Weight and Width axes to choose dynamic ranges for.

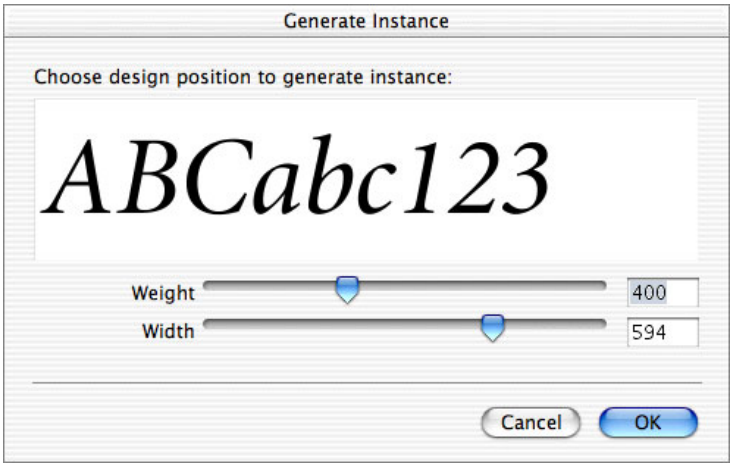
As you know, the dynamic range is the distance between the lowest and highest design coordinates, so all you have to do is to choose the proper low and high edge values.

The best way to choose these values is to compare your font with a font that already has these values assigned properly. We recommend you use one of Adobe's fonts for this comparison.

Another recommendation is to choose edge values so that the design coordinate of the Normal style (normal weight and normal width) of the font will be 500-600 units.

Generating a Single-Master Font

You can create a regular single master font from an instance of a Multiple Master font using FontLab. Use the **Tools/Multiple Master/Generate Instance** command to open the Generate Instance dialog box:

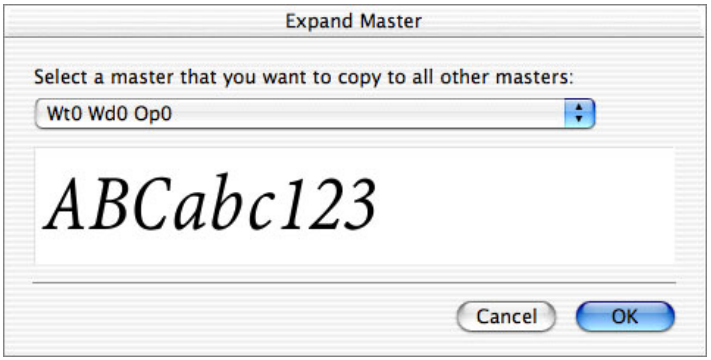


Use the controls to select the intermediate (or extrapolated) design. The Edit controls have a white background when the selection is within the dynamic range and a yellow background when extrapolation is used.

Click **OK** and FontLab will generate a single-master font and open it in a new Font Window.

Expanding the Master

Use **Tools/Multiple Master/Expand Master** to copy the contents of one of the masters to all other masters. If this command is selected when the Glyph window is active, it is applied to the currently active master. If it is used when the Font window is active, you will see a dialog box that lets you select the master for copying:



This operation may be useful if applied when the Mask layer is in editing mode – you may need to copy one of the Mask masters to all the other masters to simulate the single-master Mask layer.

Hinting Multiple Master Fonts

As you know, in FontLab you can use hints and links to do character-level hinting. You can also do character-level hinting of MM fonts using hints or links.

Hints are pairs of parallel lines that define the position and width of a vertical or horizontal stem. Hints are independent from the outline; they exist in a different layer. And they are multilayered, just like outline nodes are. If you use hints to do character-level hinting you will have to hint all the masters separately.

A better way is to use links. Links connect nodes, so they are not dependent on the number of masters in the font. On export of the MM font the links will be automatically converted to hints and hints will be automatically generated according to the master's design.

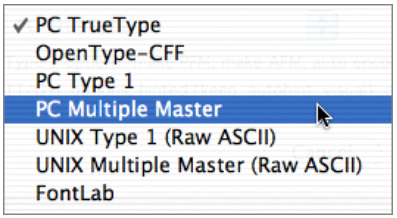
So to hint MM fonts most efficiently use vertical and horizontal links to connect the outline nodes and you will save time in direct proportion to the number of masters your font has.

You can make a hint replacement program using the Type 1 hinting tool if necessary. The preview panel of the Type 1 hinting tool will show an intermediate version of the hinted font according to the design coordinates selected in the Axis panel. Use the sliders of the Axis panel to see how your font will look on the screen when different font instances are generated.

Generating a Multiple Master Type 1 Font

When you finish your work with a MM font you need to build a font file that you can use on PostScript devices or printers.

To build a PC Multiple Master Type 1 font, choose the **File/Generate font** command. When the standard Save As dialog box appears, select PC Multiple Master format in the Format popup menu:



Press the **Save** button to save the font in PC Multiple Master format.

Note: PFB and MMM files will be saved if you make a PC MM Type 1 font. These files are necessary to install MM fonts in Adobe Type Manager in Windows.

Select Unix Multiple Master (Raw ASCII) format to make a MM font in downloadable format. Fonts in this format can be instantly downloaded to a PostScript printer.

To build a Multiple Master Type 1 font for Macintosh, choose the **File/Generate Mac Suitcase** command and refer to “Exporting for Macintosh” section.

OpenType Fonts

In this chapter we will discuss working with the OpenType fonts. The OpenType font format, jointly developed by Microsoft and Adobe, allows us to combine the best features of the TrueType and Type 1 font formats.

For the end user there is little difference between OpenType and TrueType fonts: both are stored in a single font file, both are Unicode-encoded and Windows and Mac OS directly support both.

The real difference is the support of font features that is provided by the OpenType font format. Font features, which can be script and language-dependent, can define behaviours of the font, which allow better typographic layout, and precise support of complex scripts.

For detailed OpenType specification and other documents please refer to this page:

http://www.microsoft.com/typography/otspec/default.htm

A good document covering OpenType glyph processing is:

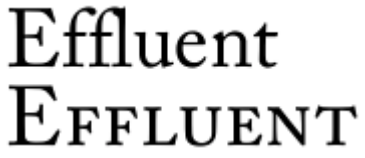
http://www.microsoft.com/typography/developers/opentype/default.htm

9

Font Features

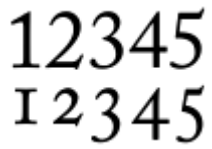
The most important thing about OpenType technology is its ability to define *font features* – rules that change the standard behaviour of the font.

Here are a few examples of such features (source text is on top):



Small caps

Ligature



Old Style Numerals

Probably the best thing about OT features is that they do not change the source string of characters. To explain this we need to again talk about the character-glyph model.

The source text that you type on the keyboard or get from another source is a sequence of characters that have strong links to the codes that the computer uses to store data. The picture of the text that you see on the screen is a sequence of glyphs or character images. It is important to understand that there is not necessarily a one-to-one relationship between character and glyph: it is possible to have a single glyph used as the image for more than one character (Latin A and Cyrillic A, for instance, are different characters, but use the same glyph) and sometimes you may have more than one glyph "serving" a single character.

Please, remember this key OpenType principle: OpenType layout engine doesn't know anything about characters! All the features that OpenType can have are defined for glyphs. This is the process of OpenType text processing:

0. As a source we have a sequence of characters.
1. Character codes are mapped to default glyphs using the Unicode mapping table. In FontLab this is what you see in the Font window when you select one of the codepages. Here we have a sequence of characters replaced by a sequence of glyphs. No character information is available beyond this point!
2. The source sequence of glyphs is passed to the OpenType processing module, which then applies the font features in a pre-defined sequence. The list of the features to apply is determined by the application (for example, in Adobe InDesign you can explicitly select features to apply) or operating system (e.g. the rendering of Arabic text with an OpenType font).
3. The resulting sequence of glyphs is passed to the second stage of feature processing which can shift the positions of glyphs. Kerning is applied at this stage.
4. The sequence of glyphs, accompanied by the positioning information, is passed to the rasterizer, which does the imaging of the features on the destination device: screen or printer.

Features and Lookups

Every feature consists of one or more *lookups*. A Lookup is an elementary procedure performed on a glyph sequence or positioning data. For instance, “replace the sequence of ‘f’ and ‘l’ with the ‘fl’ ligature glyph” is a lookup. A combination of similar lookups forms a feature.

The sequence of lookups is important. They are applied in the order they are defined. The sequence of features is also important, but the application or operating system may make changes in the feature preference.

By default all features and lookups are defined for the default language of the Latin script.

Scripts and Languages

The second great OpenType feature is support for multiple scripts and languages. With an OpenType font you can define different behaviours of the font when it is used to type text in different languages. For example, some ligatures that are necessary in English are not applicable to Turkish. Other features, like support for initial, medial and final forms of the characters are applicable only to Arabic script, and so on.

OpenType allows us to define script and language dependence at the lookup level, so the same feature may work differently when different languages are supported.

OpenType Font Formats

Another key feature of the OpenType format is the fact that from the user's point of view there is only one font format for Mac, PC or any other platform.

From the inside, there are two possible forms of OpenType fonts: so-called TrueType-flavoured (or TT-flavoured) and PostScript-flavoured (or CFF-flavoured).

The general structure of the font file is the same and both versions of the format provide the same functionality. But there are some technical differences:

Version	TT-flavored	CFF-flavored
Outlines	2nd-order, like in TrueType fonts	3rd-order, like in Type 1 fonts
Hinting	TrueType instructions	Type 1 declarative hints
File extension	TTF	OTF
Comments	Basically any TrueType font may be called a TT-flavoured OpenType font, but usually we call them OpenType fonts only when they have special features.	Outline data is stored in CFF (Compact Font Format) format that is expected to work better with PostScript devices.

What Format to Prefer

It is not easy to say which version is better. Both formats will work on both platforms. For Windows-centered office use we would recommend TT-flavoured OpenType fonts, as they will provide better compatibility with the old versions of the OS.

For cross-platform and DTP-oriented applications the CFF-flavoured version seems to have some advantage because it will provide better basic rendering quality.

Please note that the differences are minor and the most important thing to choose is the source format in which you have your fonts. If you have TrueType fonts that you want to convert to OpenType format by adding features, then the TT-flavoured format is your choice.

If you have Type 1 fonts, it will be easier to convert them to CFF-flavoured OpenType format.

OpenType Tables

OpenType fonts consist of multiple tables. Every table is identified by a tag, which is a combination of up to 4 characters.

3 tables are “responsible” for the OpenType features:

GDEF	Glyph definition table. Contains information about font glyphs, including their type (simple, mark or ligature), cursive-attachment points and position of the caret inside the ligature character.
GSUB	Glyph substitution features
GPOS	Glyph positioning features

Other tables may exist in OpenType fonts, for example the BASE table that defines different baseline positions for non-Roman scripts, but FontLab cannot work with these tables yet.

Please note that the presence of any one of these tables makes a TrueType font an OpenType font.

Feature Definition Language

Information about OpenType features is stored in a binary form inside the font file. This is not easy to modify and not easy to handle with visual tools (like the tools that FontLab provides to edit outlines that are also stored in a binary form).

To define features in human-readable form Adobe has developed the feature-definition language. It is very easy to read and it is the most compact way to represent OpenType font features.

Let's take a simple example: a ligature feature that covers the basic "fi" and "fl" ligatures that are present in almost every Western font. In feature-definition language this feature will be defined as follows:

```
feature liga{
  sub f i by fi;
  sub f l by fl;
} liga;
```

Other possible features are defined in a similar way, keeping the feature definition both compact and readable.

When FontLab opens an OpenType font file that contains features it tries to reconstruct the feature-definition file. With a few exceptions it works for most possible combinations of substitution and/or positioning features.

In the following sections we will describe the feature definition language in more detail. The next section covers the basic rules of the language.

Language Syntax

Information in this section is partially taken from the official Feature File Format specification by Adobe with their permission. Only those parts of the language that are supported by FontLab are described.

Comments

The "#" character indicates the start of a comment; the comment extends until the end of the line.

Special characters

#	pound sign	Denotes start of comment
;	semicolon	Terminates a statement
,	comma	Separator in various lists
@	at sign	Identifies glyph class names
\	backslash	Distinguishes glyph names from an identical keyword
-	hyphen	Denotes glyph ranges in a glyph class
=	equal sign	Glyph class assignment operator
'	single quote	Marks a glyph or glyph class for contextual substitution or positioning
"	double quote	Marks a glyph or glyph class for contextual substitution or positioning
{ }	braces	Enclose a feature, lookup, table, or anonymous block
[ ]	square brackets	Enclose components of a glyph class
< >	angle brackets	Enclose a device, value record, contour point, anchor, or caret
( )	parentheses	Enclose the file name to be included.

Number

A <number> is a signed decimal integer (without leading zeroes). For example:

-150
1000

Glyphs

These are represented by the glyph name. A glyph name may be up to 31 characters in length, must be entirely comprised of characters from the following set:

A-Z
a-z
0-9
. (period)
_ (underscore)

and must not start with a digit or period. The only exception is the special character ".notdef".

"twocents", "a1", and "_" are valid glyph names. "2cents" and ".twocents" are not.

An initial backslash serves to differentiate a glyph name from an identical keyword in the feature file language. For example, a glyph named "table" must be specified in the feature file as:

\table

Glyph classes

A feature file glyph class, <glyphclass>, represents a single glyph position in a sequence and is denoted by a list of glyphs enclosed in square brackets.

For example:
[endash emdash figuredash]

An example of a sequence that contains a glyph class is:

space [endash emdash figuredash] space

This would match any of the 3 sequences "space endash space", "space emdash space", or "space figuredash space" during OpenType layout.

A feature file glyph class that contains only one single glyph is known as a singleton glyph class.

A feature file glyph class is also used to represent the set of alternate glyphs in an alternate substitution lookup type rule.

Ranges

A range of glyphs is denoted by a hyphen:

[<firstGlyph> - <lastGlyph>]

Spaces around the hyphen are not required since hyphens are not permitted in feature file glyph names. For example:

[A-Z]

Named glyph classes

A glyph class can be named by assigning it to a glyph class name, which begins with the "@" character, and then referred to later on by the glyph class name. For example:

```
@dash = [endash emdash figuredash];      # Assignment
space @dash space                        # Usage
```

The part of the glyph class name after the "@" is subject to the same name restrictions that apply to a glyph name, except that its maximum length is 30.

Glyph class assignments can appear anywhere in the feature file. A glyph class name may be used in the feature file only after its definition.

When a glyph class name occurs within square brackets, its elements are simply added onto the other elements in the glyph class being defined. For example:

```
@Vowels.lc = [a e i o u];
@Vowels.uc = [A E I O U];
@Vowels = [@Vowels.lc @Vowels.uc y Y];
```

Here the last statement is equivalent to:

```
@Vowels = [a e i o u A E I O U y Y];
```

No square brackets are needed if a glyph class name is assigned to another single glyph class name. For example:

```
@Figures_lining_tabular = @FIGSDEFAULT;
```

Ranges, glyphs, and glyph class names can be combined in a glyph class. For example:

```
[zerooldstyle - nineoldstyle  ampersandoldstyle  @smallCaps]
```

In FontLab you can include FontLab classes into the list of the OpenType classes, so it will not be necessary to copy the information twice. We will discuss this later.

Including files

Including files is accomplished by the directive:

```
include(<filename>)
```

In this FontLab implementation an included file must be located in the **Features** folder.

A maximum include depth of 5 ensures against infinite include loops (files that include each other).

Specifying features

Each feature is specified in a feature block:

```
feature <feature tag> {
    # specifications go here
} <feature tag>;
```

For example:

```
feature liga {
    # ...
} liga;
```

A feature file "rule" is a statement that specifies glyph substitution or glyph positioning. A feature block may contain glyph substitution rules, glyph positioning rules, or both.

FontLab automatically separates features into feature records and provides feature templates that simplify definition of new features.

Language system

In practice, most or all of the features in a font will be registered under the same set of language systems, and a particular feature's lookups will be identical across the language systems that the feature is registered under.

The "languagesystem" statement takes advantage of this fact. It is the simplest way to specify a language system in the feature file. One or more such statements may be present in the feature file at global scope (i.e. outside of the feature blocks or any other blocks) and before any of the feature blocks:

```
languagesystem <script tag> <language tag>;
```

When these statements are present, then each feature that does not contain an explicit "script" or "language" statement will be registered under every language system specified by the "languagesystem" statement(s).

If no "languagesystem" statement is present, then the implementation will behave exactly as though the following statement were present at the beginning of the feature file:

```
languagesystem latn DFLT;
```

Script and Language

Occasionally you may need to specify a feature whose lookups vary across the language systems of the feature, or whose language systems vary from the set of language systems of the rest of the features in the file (specified by the "languagesystem" statements).

In these cases, the "script" and "language" statements should be used within the feature block itself. (A "script" and/or "language" statement must be present before the first rule in the feature in order to indicate to the feature file parser that this feature is not to be registered under the language systems specified by the "languagesystem" statements).

The feature's lookups will be registered under the script and language attributes current at the definition of the lookup. The attributes may be changed as follows:

"script" statement:

```
script <script tag>;
```

For example:

```
script kana;
```

When a "script" statement is seen, the language attribute is implicitly set to 'DFLT', and the lookupflag attribute is implicitly set to 0. The script attribute stays the same until explicitly changed by another "script" statement or until the end of the feature.

"language" statement:

The language attribute stays the same until explicitly changed, until the script is changed, or until the end of the feature. To change the language attribute, use the "language" statement:

```
language <language tag> [excludeDFLT|includeDFLT] [required];
```

The script and lookupflag attributes stay the same as before. (If no "script" assignment statement has been seen thus far in the feature block, then the script attribute is set to 'latn', but it is recommended that an explicit "script" statement be used in such cases for clarity.)

Here is an example statement:

```
language DEU;
```

As a result of this statement, (a) the language attribute is changed to 'DEU ', and (b) the 'DFLT' lookups of the current script are automatically included into the language system specified by the current script and language attributes. If (b) is not desired, as may occasionally be the case, then the keyword "excludeDFLT" must follow the language tag. For example:

```
language DEU excludeDFLT;
```

The keyword "includeDFLT" may be used to explicitly indicate the default 'DFLT' lookup-inheriting behavior. For example:

```
language DEU includeDFLT;    # Same as:    language DEU;
```

lookupflag

The chapter "Common Table Formats" in the OpenType Font File Specification describes the LookupFlag field in the Lookup table.

The lookupflag attribute defaults to 0 at the start of a feature block.

The lookupflag attribute stays the same until explicitly changed, until a lookup reference statement is encountered that changes it, until the script is changed, or until the end of the feature.

To change the lookupflag attribute explicitly, use the lookupflag statement, which takes two formats:

lookupflag format A:

```
lookupflag <named lookupflag value> (, <named lookupflag value>)*;
```

Here, the individual lookup flag values to be set are expressed in a comma-separated list of one or more <named lookupflag value>s, in no particular order. A <named lookupflag value> is one of the following:

```
RightToLeft
IgnoreBaseGlyphs
IgnoreLigatures
IgnoreMarks
```

At most one of each of the above 5 kinds of <named lookupflag value> may be present in a lookupflag statement. For example, to skip over base glyphs and ligature glyphs:

```
lookupflag IgnoreBaseGlyphs, IgnoreLigatures;
```

lookupflag format B:

```
lookupflag <number>;
```

Here the entire lookup flag value is specified simply as a <number>. The format A example above could equivalently be expressed as:

```
lookupflag 6;
```

Format A is clearly a superior choice for human readability when the lookupflag value is not 0. However, a lookupflag value of 0 can be set only with format B, not with format A:

```
lookupflag 0;
```

The base glyphs, ligatures, and marks are defined in the Glyph Properties dialog. This information is stored in the OpenType font file in the GDEF table. Please, be sure to allow export of this table in the Preferences/OpenType if you are using the lookupflag feature in the feature-definition file:

lookup

The font editor can label a set of rules and refer to it explicitly later on, in order to have different parts of the font tables refer to the same lookup. This decreases the size of the font in addition to freeing the editor from maintaining duplicate sets of rules.

To define and label a lookup, use a named lookup block:

```
lookup <label> {
    # rules to be grouped
} <label>;
```

To refer to the lookup later on, use a lookup reference statement:

```
lookup <label>;
```

For example:

```
lookup SHARED {         # lookup definition
    # ...
} SHARED;
# ...
lookup SHARED;          # lookup reference
```

Since the labelled block literally defines a single lookup in the font, the rules within the lookup block must be of the same lookup type and have the same lookupflag attribute. The lookup block must be specified within a feature block and may not contain any other kind of block.

OpenType and FontLab

Implementation of OpenType features in FontLab is based on the technologies provided by Adobe – one of the two inventors and key supporters of the format.

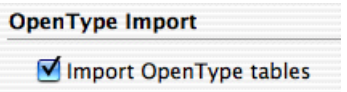
OpenType support may be separated into three stages:

1. Importing OpenType fonts and reading the features. In this stage FontLab tries to convert binary representation of the features into feature-definition file format.
2. Editing the features and previewing the results. FontLab provides a feature editor that is integrated into the FontLab user interface and a preview panel that can show how features work without actually exporting and installing the font file.
3. Feature compilation and export. At this stage an Adobe library is used to convert feature definitions into binary form and build the OpenType font files.

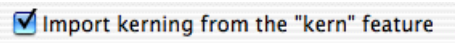
Importing OpenType Fonts

There is nothing special about importing OpenType fonts: use the **File/Open** command to open files with .ttf extension (which mean TrueType-flavoured OpenType font) or .otf extension (CFF-flavoured version of the format).

To read OpenType features make sure that this option is active in the **Application/Preferences/OpenType** page:



The following option:



will import kerning defined in the “kern” positioning feature into a font pair kerning table. We recommend that you have this feature active when you import the OpenType font.

FontLab will try to decompile OpenType features defined in the GPOS and GSUB tables. It can read almost all substitution features and single and pair positioning features. FontLab does not support Mark attachment features.

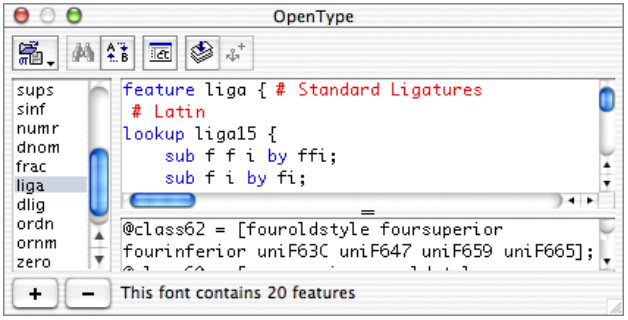
If a GDEF table is defined in the font, FontLab will read information about anchors, glyph types and caret positions from it.

OpenType Panels

FontLab provides two panels that deal with OpenType features: the OpenType panel and the OpenType Features page of the Preview panel.

OpenType Panel

In FontLab you can see OpenType features in the special OpenType panel. To open the panel use the **Window/OpenType panel** command. Below is the illustration of the panel that appears if you open the FLCaslon Adobe font (Caslon.otf):



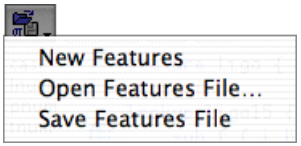
At the top of the panel you find a toolbar, at the left – a list of the features with two buttons below and a split field at the right. The top part of the field contains the feature definition text for the feature selected in the list and the bottom part contains the global definition data, which usually is a definition of the glyph classes.

You can use the separator bar between top and bottom panels to adjust their relative vertical sizes.

Buttons on the toolbar mean:

	Opens the menu. The contents of the menu are described below.
	Finds the glyph or class. When the cursor in the feature-editing panel is on the glyph name you can click on this button to quickly find the glyph in the Font or Glyph windows. **CMD+CTRL**-click on the glyph name does the same. If the cursor is on the class name FontLab will try to find the class in the bottom editing field or in the Class panel.
	Renames the glyph. Position the cursor on the glyph name and click this button to open the glyph rename dialog box, described in one of the following sections.
	Compiles the feature definitions.
	Opens the OpenType Features Preview panel.
	Adds an anchor. This button is enabled only when the Glyph window is open and active.

When you click on the button you will see a popup menu:



Commands in the menu mean:

New features	Removes all feature definition information and allows you to start making a new set of features.
Open features File	Opens an existing feature-definition file. With this command you can apply a previously-created feature-definition to different fonts. During import FontLab automatically separates features into individual records and places the class definitions in the bottom editing control.
Save features File	Saves the current set of feature definitions to a file.

Adding and Removing Features

To add a new feature click on the + button below the features list. The new feature (named “xxxx”) will appear in the list and the generic feature definition text will be posted to the feature editor:

```
feature xxxx {
   sub by ;
} xxxx;
```

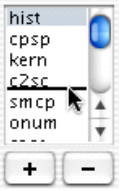
When you change the xxxx name to something real the name in the list will be updated automatically. Do not forget to make the ending name the same as the beginning name or the feature definition will not compile.

To remove the feature select it in the list and click on the − button.

Reordering Features

The order of features is sometimes important. For example, if you have the ligature feature and the small caps feature they must be applied in a pre-defined order: the small caps feature should come first and only after that may ligatures be applied.

To reorder the features simply drag-drop feature names in the list:



Entering the Glyph and Class Names

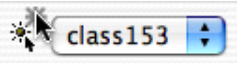
All features contain glyph names. You can enter them on the keyboard but it's faster **to use the drag-drop method**:

1. Select the glyph (in the Font window or in the Collection or in the Class panel) whose name you want to enter.
2. Press the mouse button and drag the cursor to the feature-definition field. The position where the name will be inserted is highlighted with the editing caret.
3. Select the destination position and release the mouse button to insert the glyph name.

If more than one glyph was selected for drag-drop operation, all the glyph names will be inserted separated by spaces and surrounded by brackets:

```
[zero one two three four five six seven eight nine]
```

To enter a class name defined in the Classes panel, drag the "sun-like" icon to the left of the class names popup menu:



If you drop the class name, it will be inserted into the feature definition text preceded by the '@'

```
@class153
```

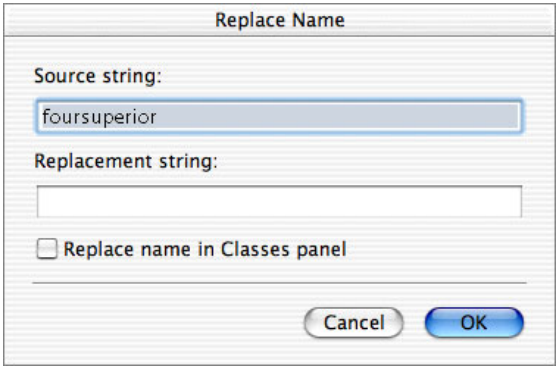
To insert the actual contents of the class hold the **Ctrl** key when you release the mouse button:

```
[perthousand perthousand percent threeeighths threequarters threeeighths
fiveeighths fiveeighths seveneighths seveneighths]
```

Renaming Glyphs and Classes

When a glyph name is entered into feature definition text the link with this glyph in the font is lost, so if you rename the glyph using the FontLab glyph rename command the feature definition will not be updated.

To rename a glyph in the feature definitions, position the cursor on the glyph name in the feature editing field and click the button on the toolbar. You will see a dialog box:



In the top field you can enter the name of the glyph you want to rename and in the bottom field you enter new name for the glyph.

Check the **Replace name in the Classes panel** option to rename the glyph in the FontLab Classes panel.

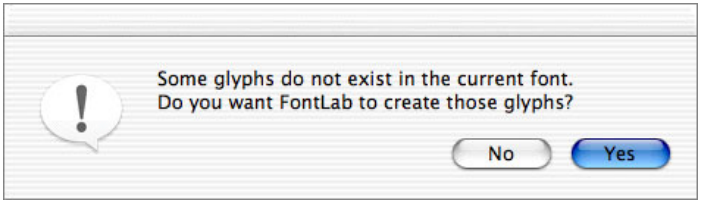
After you click **OK** in the dialog box FontLab will scan all the feature definitions and replace the old glyph name with the new one. Note that you can enter a glyph class definition as a replacement string, for instance to replace “one” by “[one two three]”.

Compiling the Feature Definitions

When you import or create OpenType features they exist as text-based definitions. To convert them to the usable binary format of the GSUB and GPOS OpenType tables they must undergo a process called “compilation”. During compilation the feature definition source code (text) is checked for errors and glyph references are verified. When everything is confirmed OK the text is converted to the binary form, which you can preview or export.

To compile feature definitions click on the button on the toolbar. FontLab will try to compile the features and if there are errors during the compilation they are put into a special Output panel.

There is one special error message: if glyphs are referenced by the feature definition but do not exist in the font, FontLab will show the warning message:



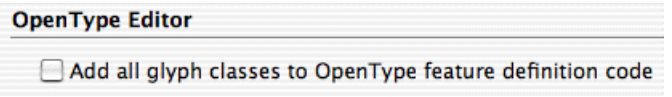
If you click **Yes**, FontLab will try to create the glyphs using their names as the source of information. The same algorithm is used in the Generate Glyphs operation described on page 345.

If you hold down the **Ctrl** key when you are clicking the button to compile the features, FontLab will compile the features and decompile the resulting binary tables as if you had imported them from an OpenType font file.

Compiling the Classes

There are two set of classes: text-based class definitions made in the feature-definition language and located in the bottom field of the OpenType panel, and FontLab glyph classes that are represented in the Classes panel (for more information about the Classes panel please refer to page 431).

It is natural to provide a link between these classes. Use the following option in the **Application/Preferences/OpenType** dialog box:

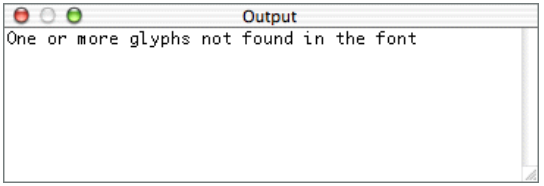


to automatically include all FontLab classes into the compiled feature-definition code.

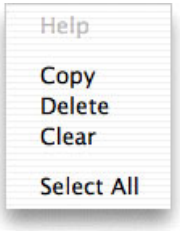
Please note that FontLab will import all OpenType classes when an OpenType font is imported and will place them in the OpenType panel and FontLab Classes panel, so when you use this option for an imported font you need to manually delete the OpenType classes **to avoid duplication**.

The Output Panel

The Output panel opens automatically when there is something to show and looks very simple:



Basically it is just a text-output window with a scroll bar. You can **Ctrl**-click the panel to open the context menu:



Use the commands in the menu to clear or copy the selected text.

You can manually open the output panel with the **Window/Output panel** command in the main FontLab menu.

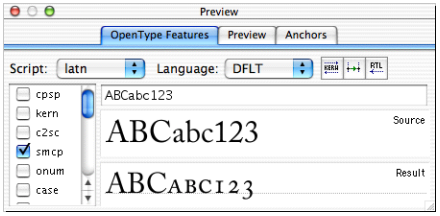
The output panel is also used by the FontLab Macro system to show the results of macro program interpretation.

OpenType Features Preview Panel

When OpenType features are compiled you can preview them with the OpenType Features page of the Preview panel.

To open the Preview panel use the **Window/Preview panel** command or click on the button located on the OpenType panel toolbar.

When there are OT features defined the OpenType Features preview panel looks like this:



In the top part of the panel you find a command bar that contains script and language popup menus and some buttons.

When more than one script or language is referred to by the feature definition, they are listed in the popup menus on the command bar. Use them to select the script and language to preview.

When a script is selected, you will see a list of the features that are defined for that script and language. Please note that when you select the non-default language, only features that are unique for the selected language are listed.

Every feature name has a check box. Use it to turn the feature on or off. When several features are selected they are applied in the order in which they are listed.

To the right of the list there are 3 fields: *source text*, *source preview panel* and *resulting preview panel*.

You must **enter the text** you want to use to preview in the source text field. Use FontLab notation to enter glyphs that are not directly accessible from the keyboard: '/' followed by the glyph name and a space or '\' followed by the Unicode index.

You can also **drag-drop glyphs** from the Font window or from the Collection or Classes panels into the source text field. Or just paste text from the Clipboard.

The **Source preview** shows the glyphs before OpenType features are applied. The **Resulting preview** panel shows the glyphs after the OpenType features do their work.

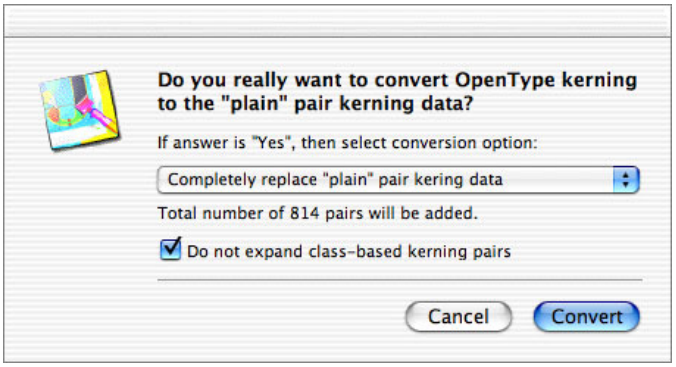
Interpretation of the features is based on the OpenType specification and is carefully tested for compatibility with OS and Adobe InDesign interpretation. However please be aware that feature interpretation is an application-centric system so it may vary in different applications.

Use the buttons on the top of the panel **to get access to options**:

	Click on this button to convert the OpenType kern feature into the plain pair kerning table. This operation is described in the following section.
	If this button is pressed, FontLab will preview the glyph metrics so you can see how the positioning features work.
	If this button is pressed both source and resulting preview windows work in right-to-left mode. This is good for working with Arabic or Hebrew fonts.

Converting the Kerning

If you have defined a kern feature you can convert it into the plain pair kerning table at any time. Just click on the button on the OpenType Features preview panel. You will see a dialog box:


Do you really want to convert OpenType kerning to the "plain" pair kerning data?
If answer is "Yes", then select conversion option:
Completely replace "plain" pair kering data
Total number of 814 pairs will be added.
Do not expand class-based kerning pairs
Cancel
Convert

There are two options that you can select in the popup menu:

Completely replace “plain” pair kerning data

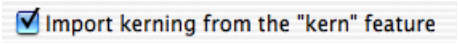
Use this option to remove existing kerning information and replace it with the result of the kern feature interpretation.

Append converted kerning to the existing kerning table

Only new pairs that do not already exist will be added to the existing pair kerning table.

Notes:

1. This operation is equivalent to the OpenType import option:

   The difference is that you can do it at any time not just during import of the OpenType font file.

2. Class-based kerning defined in the kern feature will be converted into class-based kerning as FontLab understands it. Real kerning pairs will be added only for the key glyphs. Please, refer to the Editing Metrics chapter and page 430 for more information about class-based kerning.

Feature Development Process

When you want to create a feature we recommend you follow this sequence of operations:

1. Create a feature in the OpenType panel (click on the button in the lower left corner).
2. Name the feature. Enter the name at the beginning of the feature block and at the end by replacing the default “xxxx”.
3. Fill the feature body with lookups. Description of all the types of lookups that FontLab can handle follows this section.
4. Click the button to compile the feature definition text.
5. Check the Output panel for warnings and errors. If there is an error, it will be highlighted in the feature definition text. Fix any errors.
6. When there are no errors, check the feature in the OpenType Features Preview panel. Open the panel, select the script and language and enter the test text.

At this point you know everything you need to create new features except the actual lookups that define a feature. In the following sections we will discuss the different types of substitution and positioning lookups.

Please note that you may combine substitution and positioning lookups in the same feature.

Substitution Lookups

Substitution lookups deal with the replacement of the source glyph(s) with some other glyph(s).

The simplest example of a substitution lookup is the replacement of lowercase characters by small-caps versions.

Lookups may be context-dependent or context-independent. Context-independent lookups are applied every time the source sequence of glyphs is present, like when you want to replace the ‘f’ and ‘l’ sequence with the ‘fl’ ligature. In other cases you may need to apply a substitution only when a source sequence of glyphs is surrounded by some other glyphs. For instance you may want to replace an uppercase character with a lowercase when it is followed by another lowercase character.

The OpenType specification declares the following types of basic substitutions:

Single	Replaces a single glyph with another single glyph: a->A
Ligature	Replaces multiple glyphs with a single glyph: f l -> fl
Multiple	Replaces a single glyph with multiple glyphs: $ -> d o l l a r
Alternate	Replaces a single glyph with one of the glyphs in a list: A -> A.version1 or A.version2

All these substitutions may be context independent or context-dependent.

FontLab supports all of the substitutions except the multiple substitutions.

Single Substitution

This is the simplest substitution – it replaces a single glyph with another glyph or a class of glyphs with another class. The Class form of the substitution requires that the number of glyphs in the source and destination classes be the same.

A single substitution rule is specified in one of the following formats:

```
substitute <glyph> by <glyph>;          # format A
substitute <glyphclass> by <glyph>;     # format B
substitute <glyphclass> by <glyphclass>; # format C
```

You can use the codeword “sub” instead of the longer “substitute”.

Format B specifies that all glyphs in the target glyph class will be replaced by the same replacement glyph.

Format C specifies that any of the glyphs in the target glyph class must be replaced by its corresponding glyph (in the order of glyphs in the glyph classes) in the replacement glyph class. If the replacement is a singleton glyph class, then the rule will be treated identically to a format B rule. If the replacement class has more than one glyph, then the number of elements in the target and replacement glyph classes must be the same.

For example:

```
sub a by Asmall;                                    # format A
sub [one.fitted oneoldstyle one.taboldstyle] by one; # format B
sub [a - z] by [Asmall - Zsmall];                   # format C
sub @Capitals by @CapSwashes;                       # format C
```

The third line in the above example produces the same effect in the font as:

```
sub a by Asmall;
sub b by Bsmall;
sub c by Csmall;
# ...
sub z by Zsmall;
```

Sample applications of this rule:

Replace different types of figures:

```
sub @ordfigures by @propfigures;
```

Replace lowercase glyphs by small-caps:

```
sub @lowercase by @smallcaps;
```

Ligature Substitution

The ligature substitution rule replaces several glyphs in sequence with a single glyph.

A Ligature substitution rule is specified as:

substitute <glyph sequence> **by** <glyph>;

<glyph sequence> must contain two or more <glyph|glyphclass>es. For example:

substitute [one oneoldstyle] [slash fraction] [two twooldstyle] **by** onehalf;

Since the OpenType specification does not allow ligature substitutions to be specified on target sequences that contain glyph classes, the implementation software will enumerate all specific glyph sequences if glyph classes are detected in <glyph sequence>. Thus, the above example produces the same effect in the font as if the font editor manually enumerated all the sequences:

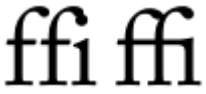
substitute one slash two **by** onehalf;
substitute oneoldstyle slash two **by** onehalf;
substitute one fraction two **by** onehalf;
substitute oneoldstyle fraction two **by** onehalf;
substitute one slash twooldstyle **by** onehalf;
substitute oneoldstyle slash twooldstyle **by** onehalf;
substitute one fraction twooldstyle **by** onehalf;
substitute oneoldstyle fraction twooldstyle **by** onehalf;

Almost all fonts contain at least two ligatures: “fl” and “fi” which can be easily encoded as:

```
substitute f l by fl;
substitute f i by fi;
```

Some fonts add longer ligatures:

```
substitute f f i by ffi;
```



A contiguous set of ligature rules does not need to be ordered in any particular way by the font editor; the implementation software does the appropriate sorting. So:

```
sub f f by ff;
sub f i by fi;
sub f f i by ffi;
sub o f f i by offi;
```

will do the same thing as:

```
sub o f f i by offi;
sub f f i by ffi;
sub f f by ff;
sub f i by fi;
```

Alternate Substitution

Alternate substitution replaces a glyph with one of the glyphs in a predefined list of alternatives. The application that uses the font is expected to decide which glyph to choose. A good example of this lookup is to provide several versions of some glyph, like the ampersand. Another application is the selection of several different forms of ornaments.

An alternate substitution rule is specified as:

substitute <glyph> **from** <glyphclass>;

For example:

substitute ampersand **from** [ampersand.1 ampersand.2];

or ornament variations:

substitute asterisk **from** [orn.1 orn.2 orn.3 orn.4];

Context-Dependent Substitutions

A context-dependent rule can be any of the rules described above with one important difference: it defines a context that must include a target sequence of glyphs (or glyph classes).

In the simple form of, say, ligature substitution we simply write:

```
sub a b c by D;
```

In context-dependent substitution we can declare that “abc”, which is a target sequence of glyphs for a ligature substitution rule, must be a part of a larger context:

```
sub period a’ b’ c’ period by D;
```

Only when “abc” is surrounded by two “period” glyphs will substitution take place. Note that we have marked the target glyphs with the single quote character positioned immediately after the glyph name.

The rule is specified as follows:

```
substitute <marked glyph sequence>   # Target sequence with marked glyphs
by <glyph sequence>;                 # Sub-run replacement sequence
```

A <glyph sequence> comprises one or more glyphs or glyph classes.

<marked glyph sequence> is a <glyph sequence> in which a set of glyphs or glyph classes is identified, i.e. "marked". We will call this marked set of glyphs a *sub-run*. A sub-run is marked by inserting a single quote (') after each of its member elements.

This sub-run represents the target sequences of the lookups called by this rule. The lookup type of the lookup called by this rule is auto-detected from their target and replacement sequences in the same way as in their corresponding stand-alone (i.e. non-contextual) statements.

Example 1. This calls a lookup. The rule below means: in sequences "a d" or "e d" or "n d", substitute "d" by "d.alt".

```
substitute [a e n] d' by d.alt;
```

Example 2. This also calls a *single substitution* lookup. The rule below means: if a capital letter is followed by a small capital, then replace the small capital by its corresponding lowercase letter.

```
substitute [A-Z] [Asmall-Zsmall]' by [a-z];
```

Example 3. This calls a *ligature substitution* lookup. The rule below means: in sequences "e t c" or "e.begin t c", substitute the first two glyphs by the ampersand.

```
substitute [e e.begin]' t' c by ampersand;
```

Specifying Exceptions to the Context Rule

Exceptions to a chaining contextual substitution rule are expressed by inserting a statement of the following form anywhere before the chaining contextual rule and in the same lookup as it:

ignore substitute <marked glyph sequence> (, <marked glyph sequence>)*;

The keywords "ignore substitute" are followed by a comma-separated list of <marked glyph sequence>s. At most one sub-run of glyphs or glyph classes may be marked in each <marked glyph sequence>, by a single-quote (') following each glyph or glyph class. This marked sub-run, when present, is taken to correspond to the "input sequence" of that rule. This generally means that it should correspond to the place where substitution would have occurred had the sequence not been an exception (see examples below). This is necessary for the OpenType layout engine to correctly handle skipping this sequence. When no glyphs are marked, then only the first glyph or glyph class is taken to be marked.

The "ignore substitute" statement works by creating subtables in the GSUB that tell the OT layout engine simply to match the specified sequences, and not to perform any substitutions on them. As a result of the match, remaining rules (i.e. subtables) in the lookup will be skipped.

Example 1. Ignoring specific sequences:

The "ignore substitute" rule below specifies that the substitution in the "substitute" rule should not occur for the sequences "f a d", "f e d", or "a d d". Note that the marked glyphs in the exception sequences indicate where a substitution would have occurred; this is necessary for the OpenType layout engine to correctly handle skipping this sequence.

```
ignore substitute f [a e] d', a d' d;
substitute [a e n] d' by d.alt;
```

Example 2. Matching a beginning-of-word boundary:

```
ignore substitute @LETTER f' i';
substitute f' i' by f_i.begin;
```

The example above shows how a ligature may be substituted at a word boundary. @LETTER must be defined to include all glyphs considered to be part of a word. The substitute statement will get applied only if the sequence doesn't match "@LETTER f i"; i.e. only at the beginning of a word.

Example 3. Matching a whole word boundary:

```
ignore substitute @LETTER a' n' d', a' n' d' @LETTER;
substitute a' n' d' by a_n_d;
```

In this example, the a_n_d ligature will apply only if the sequence "a n d" is neither preceded nor succeeded by a @LETTER.

Example 4. This shows a specification for the contextual swashes feature:

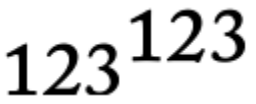
```
feature cswh {
 # --- Glyph classes used in this feature:
 @BEGINNINGS = [A-N P-Z Th m];
 @BEGINNINGS_SWASH = [A.swash-N.swash P.swash-Z.swash T_h.swash m.begin];
 @ENDINGS = [a e z];
 @ENDINGS_SWASH = [a.end e.end z.end];

 # --- Beginning-of-word swashes:
 ignore substitute @LETTER @BEGINNINGS';
 substitute @BEGINNINGS' by @BEGINNINGS_SWASH;

 # --- End-of-word swashes:
 ignore substitute @ENDINGS' @LETTER;
 substitute @ENDINGS' by @ENDINGS_SWASH;
} cswh;
```

If a feature targets only glyphs at the beginning or ending of a word, such as the 'init' and 'fina' features, then the application could be made responsible for detecting the word boundary; the feature itself would be simply defined as the appropriate substitutions without regard for word boundary.

Positioning Lookups

The OpenType specification allows you to define many positioning lookups. The lookup types may be separated into three groups:

1. Basic lookups, single and pair positioning:



2. The Cursive attachment lookup that allows smooth connection of script and cursive glyphs:

3. Mark attachment lookups that define relative positions of glyphs and marks:

FontLab 4 can support only the first group of lookups: single and pair positioning. We expect to support cursive and mark attachment lookups in one of the next releases of FontLab.

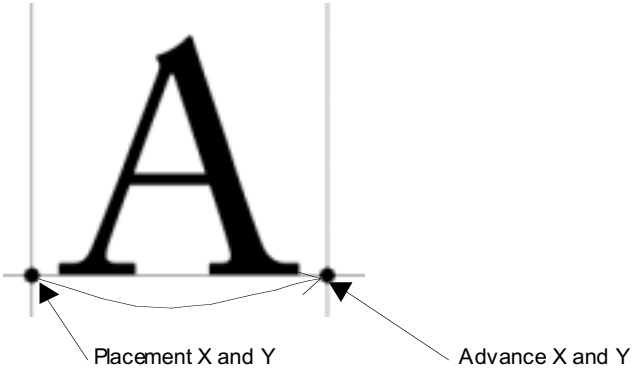
As in the case with substitution, positioning lookups may be *context-free* and *context-dependent*. Context-dependent lookups are the same as context-free but add a context that is verified against the source sequence of glyphs. Only when the context is matched is positioning performed.

Please note that glyph positioning is performed **after** substitution and that all positioning lookups must be defined for the glyph string that is a result of substitution.

Glyph positioning rules begin with the keyword "position"; this keyword may be abbreviated as "pos". (The "enumerate" or "ignore" keywords may precede the "position" keyword in some cases.) The GPOS lookup type is auto-detected from the format of the rest of the rule.

Glyph Geometry

Positioning lookups may change one of the glyph positioning metrics:



A single positioning lookup may tweak any of four values: placement_X, placement_Y, advance_Y and advance_Y.

Modification of the origin point of the glyph will shift it and all following glyphs. Modification of the advance vector will shift the next glyph in the glyph string.

Value Record

A <valuerecord> is used in positioning rules to define offsets to shift glyph origin or advance vector. It must be enclosed by angle brackets, except for format A, in which the angle brackets are optional. Note that the <metric> adjustments indicate values (in design units) to add to (positive values) or subtract from (negative values) the placement and advance values provided in the font (in the 'hmtx' and 'vmtx' tables).

Value record format A:

```
< <metric> >        # Angle brackets are optional
```

Here the <metric> represents an X advance adjustment, except when used in the 'vkrn' feature, in which case it represents a Y advance adjustment. All other adjustments are implicitly set to 0. This is the simplest feature file <valuerecord> format, and is provided since it represents the most commonly used adjustment (i.e. for kerning). For example:

```
-3          # without <>
<-3>        # with <>
```

Value record format B:

```
< <metric> <metric> <metric> <metric> >
```

Here, the <metric>s represent adjustments for X placement, Y placement, X advance, and Y advance, in that order. For example:

```
<-80 0 -160 0>        # X placement adj: -80; X advance adj: -160
```

Single Positioning

A Single Pos rule is specified as:

```
position <glyph|glyphclass> <valuerecord>;
```

Here, the <glyph|glyphclass> is adjusted by the <valuerecord>. For example, to reduce the left and right sidebearings of a glyph each by 80 design units:

```
position one <-80 0 -160 0>;
```

To shift the glyph up by 100 units:

```
position A <0 100 0 -100>;
```

Note that we changed the placement by 100 units but compensated for that with a –100 change applied to advance. This is needed to shift only the current glyph and not the following glyphs in the string.

Pair Positioning

Rules for this lookup type are usually used for kerning and must follow this format:

position <glyph|glyphclass> <glyph|glyphclass> <valuerecord format A>;

This format is provided since it closely parallels the way kerning is expressed in a plain pair kerning table. Here, the <valuerecord> must be of value record format A only, and corresponds to the first <glyph|glyphclass>.

Kerning can most easily be expressed with this format. This will result in adjusting the first glyph's X advance, except when in the 'vrkn' feature, in which case it will adjust the first glyph's Y advance. Some examples:

```
pos T a -100;          # specific pair (no glyph class present)
pos [T] a -100;        # class pair (singleton glyph class present)
pos T @a -100;         # class pair (glyph class present, even if singleton)
pos @T [a o u] -80;    # class pair
```

Note that if at least one glyph class is present (even if it is a singleton glyph class), then the rule is interpreted as a class pair; otherwise, the rule is interpreted as a specific pair.

In the 'kern' feature, the specific glyph pairs will typically precede the glyph class pairs in the feature file, mirroring the way that they will be stored in the font.

```
feature kern {
    # specific pairs for all scripts
    # class pairs for all scripts
 } kern;
```

Enumerating Pairs

If some specific pairs are more conveniently represented as a class pair, but the editor does not want the pairs to be in a class kerning subtable, then the class pair must be preceded by the keyword "enumerate" (which can be abbreviated as "enum"). The implementation software will enumerate such pairs as specific pairs. Thus, these pairs can be thought of as "class exceptions" to class pairs. For example:

```
@Y_LC = [y yacute ydieresis];
@SMALL_PUNC = [comma semicolon period];

enum pos @Y_LC semicolon  -80;      # specific pairs
pos f quoteright 30;                # specific pair
pos @Y_LC @SMALL_PUNC -100;         # class pair
```

The enum rule above can be replaced by:

```
pos y semicolon -80;
pos yacute semicolon -80;
pos ydieresis semicolon -80;
```

without changing the effect on the font.

Subtable Breaks

The implementation software will insert a subtable break within a run of class pair rules if a single subtable cannot be created due to class overlap. A warning will be given. For example:

```
pos [Ygrave] [colon semicolon] -55;          # [line 99]   In first subtable
pos [Y Yacute] period -50;                   # [line 100]  In first subtable
pos [Y Yacute Ygrave] period -60;            # [line 101]  In second subtable
```

will produce a warning that a new subtable has been started at line 101, and that some kern pairs within this subtable may never be accessed. The pair (Ygrave, period) will have a value of 0 if the above example comprised the entire lookup, since Ygrave is in the coverage (i.e. union of the first glyphs) of the first subtable.

Sometimes the class kerning subtable may get too large. The editor can force subtable breaks at appropriate points by inserting the statement:

```
subtable;
```

between two class kerning rules. The new subtable created will still be in the same lookup, so the editor must ensure that the coverages of the subtables thus created do not overlap. For example:

```
pos [Y Yacute] period -50;                   # In first subtable

subtable;                                    # Force a subtable break here

pos [A Aacute Agrave] quoteright -30;        # In second subtable
```

If the subtable statement were not present, both rules would be represented within the same subtable.

Known Features

OpenType feature processing is an application-centric system. The application knows which features it needs to apply and then searches the font for these features. If a feature is present in the font, it is applied. To implement this system features must be standardized and registered so everybody will know what a feature with a particular name does.

The Microsoft Typography group performs feature registration:

http://www.microsoft.com/typography

The full list of registered features may be found in this document:

OpenType Layout tag registry
http://www.microsoft.com/typography/otspec/ttoreg.htm

This document contains descriptions of all name tags that can be used in an OpenType font for script, language and feature names.

We will provide brief definitions of the most commonly used features.

init, medi, fina and isol Features

These features are most commonly used in Arabic scripts and allow one to define four forms of Arabic characters: initial, final, medial and isolated:

Isolated **Final** **Medial** **Initial**

Note that Arabic text is written right-to-left.

Please note that it is the application or the operating system that detects word boundaries and applies one of the features. You are not expected to pay attention to that in the feature definition.

Usually these features are simple substitutions:

```
feature init{
  sub @isolated_forms by @initial_forms;
}init;
```

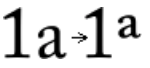
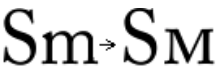
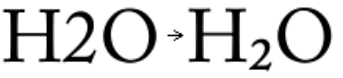
Latin Features

cpsp	Capital Spacing	Globally adjusts inter-glyph spacing for all-capital text.
		`feature cpsp {`
		`  pos @uppercase <7 0 14 0>;`
		`} cpsp;`

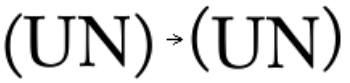
pnum	Proportional Figures	Replaces figure glyphs set on uniform (tabular) widths with corresponding glyphs set on glyph-specific (proportional) widths.
		`feature pnum {`
		`  sub @figures by @figures_prop;`
		`} pnum;`

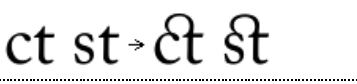
lnum	Lining Figures	This feature changes selected figures from oldstyle to the default lining form

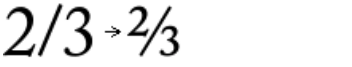
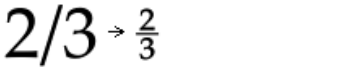
hist	Historical Forms	Replaces glyphs with their historical forms, like long form of s or Fraktur form of k.
		`feature hist {`
		`  sub s by longs;`
		`} hist;`

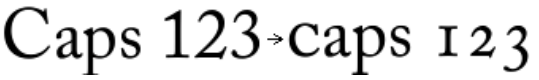
ordn	Ordinals	Replaces default alphabetic glyphs with the corresponding ordinal forms for use after figures. 
smcp	Small Capitals	This feature replaces lowercase characters with small capitals 
sinf	Scientific Inferiors	Replaces lining or oldstyle figures with inferior figures (smaller glyphs which sit lower than the standard baseline, primarily for chemical or mathematical notation). 
ornm	Ornaments	This feature lets the user access ornament glyphs in the font.

liga	Standard Ligatures	Replaces a sequence of glyphs with a single glyph, which is preferred for typographic purposes. This feature covers the ligatures that the designer/manufacturer judges should be used in normal conditions.
		`feature liga {`
		`  sub f f i by ffi;`
		`  sub f i by fi;`
		`  sub T h by T_h;`
		`  sub f f j by f_f_j;`
		`  sub f f l by ffl;`
		`  sub f f by ff;`
		`  sub f j by f_j;`
		`  sub f l by fl;`
		`} liga;`

case	Case-Sensitive Forms	Shifts various punctuation marks up to a position that works better with all-capital sequences or sets of lining figures; also changes oldstyle figures to lining figures. 

dlig	Discretionary Ligatures	Replaces a sequence of glyphs with a single glyph, which is preferred for typographic purposes. This feature covers those ligatures that may be used for special effect, at the user's preference.
		`feature dlig {`
		`  sub c t by c_t;`
		`  sub s t by uniFB06;`
		`  sub longs h by longs_h;`
		`  sub longs i by longs_i;`
		`  sub longs l by longs_l;`
		`  sub longs t by uniFB05;`
		`  sub longs longs by longs_longs;`
		`} dlig;`
		 

frac	Fractions	Replaces figures separated by a slash with 'common' (diagonal) fractions. 
afrc	Alternative Fractions	Replaces figures separated by a slash with an alternative form. 
dnom	Denominators	Replaces selected figures that follow a slash with denominator figures.

c2sc	Small Capitals From Capitals	This feature replaces capital characters with small capitals. 
numr	Numerators	Replaces all with numerator figures.
onum	Oldstyle Figures	This feature changes selected figures from the default lining style to oldstyle form.
sups	Superscript	Replaces lining or oldstyle figures with superior figures and replaces lowercase letters with superior.
sinf	Scientific Inferiors	Replaces lining or oldstyle figures with inferior figures.

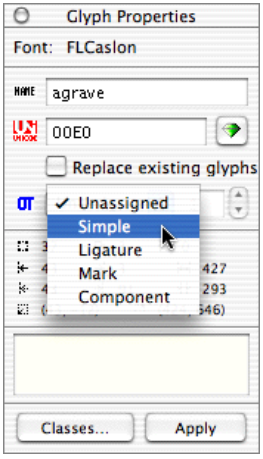
OpenType Glyph Properties

The OpenType format allows you to define a glyph's type. This can be one of the following values:

Unassigned	Glyph type is not defined
Simple	Single character
Ligature	Multiple characters
Mark	Non-spacing glyph used as a mark
Component	Part of a character

Glyph type is used in a lookup when you need to limit it to the glyphs of a particular type using the **lookupflag** command (page 644).

You define the glyph type using the glyph properties panel. Select one or more glyphs in the Font window and open the panel with the **Edit/Properties** command:



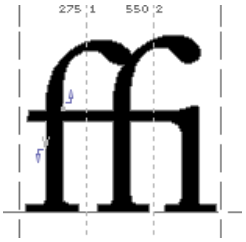
Select the glyph class in the popup to the right of the OT icon. If multiple glyphs were selected, the new type will be assigned to all of them.

Caret Positioning

When several characters are replaced by a ligature glyph using the ligature substitution lookup, a text-layout application may keep the positioning cursor inside a ligature as if it still was a set of characters:

With FontLab you can define caret positions inside the ligature. **Do the following:**

1. Open the ligature glyph in the Glyph window.
2. Open the properties panel with the **Edit/Properties** command.
3. In the properties panel select the ligature glyph type.
4. To the right of the glyph type popup menu enter the number of components in the ligature.
5. Make sure that **View/Show Layers/Anchors and Carets** option is on.
6. You will see several vertical caret lines appeared in the Glyph window. The number of the lines equals the number of components of the ligature minus one:



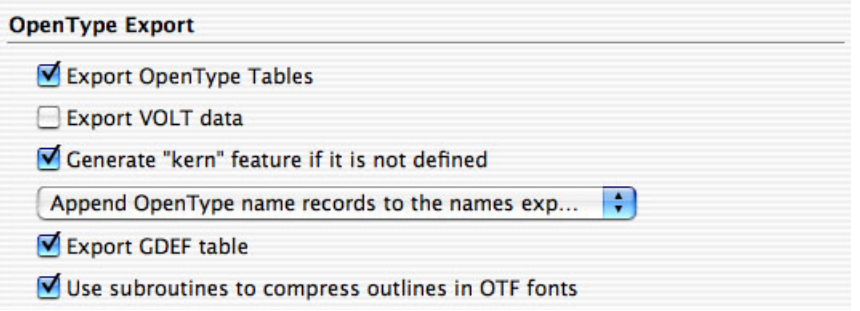
Use the Edit tool to move the caret lines to the desired positions.

Exporting OpenType Fonts

When you have a font that contains feature definitions and you want them to be exported to an OpenType font file, the first thing you need to do is to compile and test your features using the tools in the OpenType panel.

If feature definitions contain errors, they will not be exported and you will get a plain TrueType font file without any features. If you are exporting a CFF-flavoured OpenType font and the feature definitions contain an error the font will not be exported at all.

When you know that there are no errors in the features and the features work as expected, you need to check the OpenType export options in the **Application/Preferences/OpenType** dialog box:


OpenType Export
Export OpenType Tables
Export VOLT data
Generate "kern" feature if it is not defined
Append OpenType name records to the names exp...
Export GDEF table
Use subroutines to compress outlines in OTF fonts

Export OpenType Tables

OpenType features will be exported only when this option is active.

Export VOLT data

If this option is switched on, FontLab will try to export feature definition data in a format supported by Microsoft VOLT (Visual OpenType Layout Tool). Not all features can be exported in this format. Refer to the next section for more information.

Generate “kern” feature if it is not defined

If the font contains a pair kerning table but no “kern” feature and this option is switched on, FontLab will automatically generate the kern feature so programs that need this feature to make kerning will work correctly.

Export GDEF table

Turn on this feature to save information about the glyph types and caret positions to the font file. Note that this is a requirement if you have used the lookupflag operator in feature definitions.

After all options are set correctly, use the **File/Generate Font** command **to save the OpenType font file**. Select **PC TrueType (*.ttf)** to save a TrueType-flavoured OpenType font or **OpenType-CFF (*.otf)** to save a CFF-flavoured font.

FontLab and VOLT

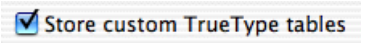
Microsoft VOLT (Visual OpenType Layout Tool) is an OpenType feature editor program developed and supported by the Microsoft Typography group. The big benefit of this program is that it can support all the features of OpenType including those that FontLab cannot handle. VOLT is based on a completely different user interface and provides visual tools to define substitution and positioning lookups.

You can get more information about VOLT at this location:

http://www.microsoft.com/typography/developers/volt/default.htm

VOLT stores OpenType features and lookups information in a text-based format which, during work on a font, is saved in special tables in a TrueType font. When work on the font is done, these tables are compiled to GPOS, GDEF and GSUB tables and intermediate tables are stripped from the font file.

If this option in the **Application/Preferences/TrueType**:



is switched on, you can open the saved font in VOLT and it will contain the special tables that VOLT uses to store intermediate data and on export these tables will be restored unchanged. This means that even if you start work on a font in VOLT, but then realise that some glyphs must be modified, you can save the font in VOLT, open it in FontLab, modify the glyphs and return to VOLT to continue your work on OpenType features.

Another FontLab feature is its ability to save simple lookups and some glyph information in VOLT format so you can start working on a font in FontLab then continue in VOLT.

You can save font features defined in FontLab in two ways:

1. Use the button in the OpenType panel to open a menu and then select the **Save Features** command to open a File Save dialog box. Select **VOLT project files** (*.vtp) as the destination format and FontLab will try to save the features in VOLT format.
2. Turn on the **Export VOLT data** option in the **Application/Preferences/OpenType** dialog and save the font as a TrueType-flavoured OpenType (PC TrueType). FontLab will write both binary OpenType tables and tables that VOLT can read so you can open this font in VOLT on PC and continue editing features.

Option 2 is the only way to open the OpenType font containing the features for editing in VOLT. By itself it cannot decompile binary OpenType tables.

Macro Programming

One of the unique features of FontLab 4 is an integrated macro programming language. With this feature you can program repeated tasks, define custom font transformations, create your own editing tools, integrate FontLab into a font development system which may include other tools, and use FontLab in many other powerful ways.

Macro programs in FontLab are written in the well-known and well-documented Python programming language. FontLab uses the standard version of the language so almost all macros written in Python will work in FontLab. In addition to support of Python FontLab provides a detailed set of classes and variables that open all the FontLab data structures to the programming interface.

Please note that this chapter covers only the very basic features of FontLab macro programming. We will be providing more information, specifications and sample programs on our web site (www.fontlab.com).

10

The Python Programming Language

Python is a very high level object-oriented programming language. It combines a very clear and easy-to-understand syntax with great power, flexibility and extensibility.

Python works on all known platforms and is intensively maintained and updated by many professionals around the world.

It is not surprising that during the last few years Python has become a de-facto standard for macro programming related to fonts. FontLab continues this trend and extends it to a new level – providing full integration of macro programming tools with its user interface.

More information about Python programming, manuals and samples are available on the official site:

http://www.python.org

which we recommend highly if you are not already familiar with the language. We will provide minimal instruction in Python programming in this chapter as that is better obtained elsewhere. We will assume that if you plan to write FontLab macro programs that you have read the Python tutorials and have some experience in Python programming.

Installing Python

When you run FontLab 4.5 for the first time it will know nothing about macro programming and Python. All Python-related tools and interface components are hidden and disabled. This means that if you don't want to use these tools you are not required to and FontLab will work smoothly without any Python integration.

If you want to use macro programs or perhaps create some programs yourself, however, you have to install the Python interpreter.

FontLab 4.5 can work with only one version of the Python interpreter: version 2.2.1. First, download it from the FontLab site (http://www.fontlab.com/html/fontlab.html provides a download link) or from the Python site:

http://python.org/ftp/python/2.2.1/MacPython221full.hqx

After you download this file, run it and follow the instructions to install Python. By default the installer places the "Python 2.2.1" folder in the Applications folder under Mac OS X. Make sure you have Admin rights to install there.

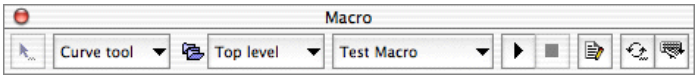
Please note that even if you have a newer version of Python installed it is a requirement to install version 2.2.1 to run Python from inside FontLab 4.5.

Macro Toolbar

You can use macro programs without any programming. FontLab includes some sample programs in its basic installation and more programs are available from our web site and from other sources.

The easiest way to run a macro program is to use the Macro toolbar. To open the toolbar click: **View/Toolbars/Macro**. If this command is disabled it means that the Python system is not installed. Exit FontLab; follow the instructions above to install Python 2.2.1 and then run FontLab again to activate the Macro features.

When the Macro toolbar appears it looks like this:



Let's describe the contents of the toolbar from left to right:

Python tool selector. A Python tool is an editing tool (like Edit, Meter or Sketch) which has all its functions defined in Python. I.e. a FontLab tool written in Python.

Macro program selectors. The left popup menu allows you to select one of the categories of programs and the right popup menu selects the program within the category.

Runs the program selected in the popup described above.

Stops the program that is running.

Opens the currently selected program in the Edit Macro Panel for editing.

Restarts the Python system. Use this command to reload all standard libraries and free all memory allocated by the Python system.

Assigns the currently selected program to a key combination.

Let's try to run one of the sample programs supplied with FontLab:

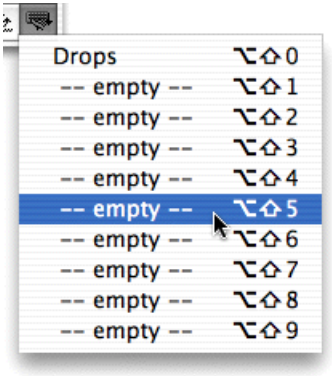
1. Open the 'B' glyph in a Glyph Window.
2. Open the Macro toolbar.
3. Select **Effects** in the folders popup menu and **Drops** in the macro popup menu.
4. Click on the ▸ button to run the program.

This is what you should see as a result of the transformation:

Assign to Keyboard

Using the Macro toolbar you can assign up to 10 macro programs to the keyboard combinations **Shift+Option+0**, **Shift+Option+1**, and up to **Shift+Option+9**. To do so:

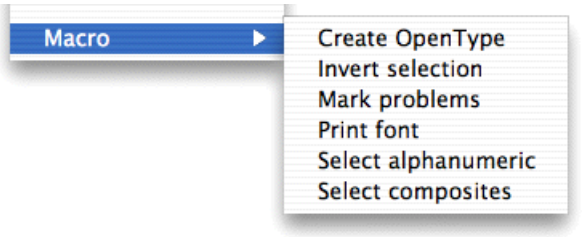
1. Select the macro program using the two popup menus on the toolbar.
2. Click on the button and select a combination in the popup menu:



The next time you press the key combination FontLab will run the assigned macro program.

Integrating into Menus

You can integrate macro programs (supplied with FontLab, written by you or downloaded from the Internet) into many context menus in FontLab. If a menu has macro programs assigned they appear in the Macro submenu at the bottom:



Integration of macro programs into menus is done automatically when macro programs are stored in one of the folders in the **FontLab/Macros/System** folder. Note that Python macro programs must have a ".py" extension to be accepted by FontLab.

Folder name	Description of the associated menu
Bitmap	Context menu when the bitmap background manipulating tool is active
Component	Context menu when the component editing tool is active
Font	Font window context menu
FontsList	Popup menu that appears when you click on the button in the Fonts panel
Glyph	Glyph window context menu
Metrics	Metrics window context menu
Node	Node context menu
Selection	Selection (in the Glyph window) context menu
TTH	TrueType hinting tool context menu.

Macro Tool

With FontLab you can define special editing tools that are entirely written in Python. By default FontLab provides 3 sample tools: Curve, Line and Drops. To activate a tool:

1. Open a glyph for editing in the Glyph window.
2. Select the tool in the leftmost list of the Macro toolbar.
3. Click on the button on the toolbar to activate a tool.

Activate one of the standard editing tools (on the Tools toolbar) to finish using the macro tool.

The default macro tools do the following:

Line tool draws lines. Click the mouse button anywhere and drag the mouse to draw a line. If the nodes are visible (**View/Show Layers/Nodes**) you can continue defining a new contour by drawing a new line from the end node of the previous line.

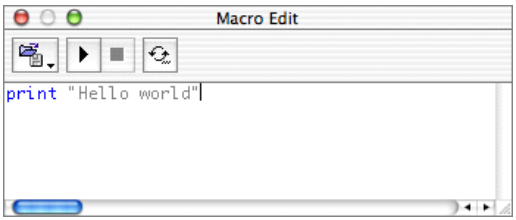
Curve tool does the same but draws curves. It emulates the curve drawing process that is common for Macromedia Fontographer® and Freehand®.

Drop tool – just a whimsical example – when you drag the mouse it adds a series of filled circles of random radius.

We will not document the process of creating new tools here. It is a relatively complex programming job. If you are interested, please visit our website for more technical documents related to FontLab macro programming.

Edit Macro Panel

If you feel ready to create your first macro program you can start by opening the Edit Macro panel. Use **Window/Edit Macro panel** to make it visible:



This is basically a simple text editing control with a toolbar at the top. The buttons on the toolbar mean:

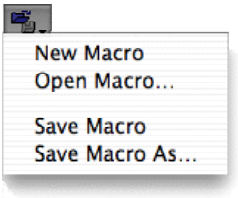
Opens the menu that contains New, Open and Save operations

Runs the currently edited program

Stops the program that is running

Restarts the Python system. Use this command to reload all standard libraries and free all memory allocated by the Python system.

When you click the button you will see a menu:



The commands in this menu let you perform standard file operations on your current macro program.

Naming the Programs

When you are saving a program and want it to be used in the Macro toolbar or to be integrated into one of the FontLab menus you need to name it. There are two ways to name a program: you can store the name in the file name (followed by the ".py" extension) or you can embed the name into the program code. The latter way is recommended – it allows you to keep the filename small but descriptive.

To name a program put the following line at the very beginning of the program:

#FLM: <program name>

where the <program name> is the name of the program, as in the following example:

#FLM: Shadow Effect

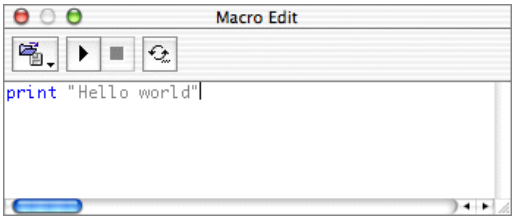
Note that there is exactly one single space between #FLM: and the name.

First Steps

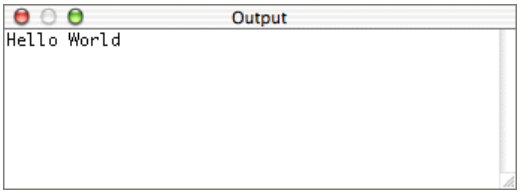
Let's write a few basic programs. The "Hello World!" program is a typical benchmark of the simplest useful program you can write. It is very easy to do in FontLab/Python:

1. Open the Edit Macro panel.
2. If there is something there, click on the button and select the New Macro command to clean the editing field.
3. Enter the following code:

```
print "Hello World"
```



4. Click on the button to run the program.
5. You will see the output panel appear on the screen containing our text:



All text you output with the Python print operation appears in the Output panel. There is more information about using this panel in the "OpenType Fonts" chapter on page 657.

OK, now let's do something more useful using FontLab classes (which are partially described below). Suppose we want to find all the glyphs that are empty: i.e. don't have an outline or any components. The "space" is a good example of such a glyph.

First, open the font that you want to check and open and clear the Edit Macro panel.

Type in the following simple program:

```
for g in fl.font.glyphs:
  if len(g) == 0 and len(g.components) == 0:
    print g.name
```

Run the program and check the results in the Output panel.

Let's talk a little about the code above:

```
for g in fl.font.glyphs:
```

This line starts a loop that assigns the variable "g" to each of the glyphs contained in the current font (which is referenced as "fl.font").

```
if len(g) == 0 and len(g.components) == 0:
```

This line compares the length of the glyph outline and the length of the "components" array (which contains all components) with zero and if there are neither outline nor components it executes the next line.

```
print g.name
```

If the condition given in the second line is true, this line prints the glyph name.

As you can see, with only 3 lines of code we have solved a problem that typically takes about 30 minutes to perform.

FontLab Python Classes

In this section we will discuss a basic set of FontLab classes and variables. The full specification is available as a separate document for download from our site (www.fontlab.com) and most classes are also self-documented. For example, to print a short reference of the class Font, type in the Edit Macro panel:

```
print Font().__doc__
```

And check the Output panel for the reference. Note the two underscore characters before and after the “doc”.

FontLab

The highest class in the FontLab hierarchy is a class named FontLab. You cannot create it explicitly, but the object of this class is always available and is named “fl”.

This class contains seven most important members:

ifont	Index of the active font
font	Current font as a Font object
ifontslist	Index of currently selected font in the fonts list panel
iglyph	Index of the currently active glyph in the current font
glyph	Current glyph as a Glyph object
count	Read-only – number of opened fonts
count_selected	Number of selected glyphs in the Font Window.

You can also access the fl object as a list of Font-type objects:

```
print fl[0].font_name
```

There are several important methods:

Close(fontindex)	Closes the current or 'fontindex' font
Open(filename) **Open(filename, addtolist)**	Opens the font from the file using the current opening options. If 'addtolist' is True, the font is added to FontLab's font list
Save(filename) **Save (fontindex, filename)**	Saves the current or selected font using the standard FontLab Save routine
Add(font)	Adds 'font' to the list of open fonts and opens the Font Window for it
UpdateFont() **UpdateFont(fontindex)**	Updates the current font or 'fontindex' (slow operation)
UpdateGlyph() **UpdateGlyph (glyphindex)**	Updates the current or 'glyphindex' glyph of the current font
EditGlyph() **EditGlyph (glyphindex)**	Opens a Glyph window for the 'glyphindex' glyph in the current font
Selected() **Selected(glyphindex)**	Returns True if the current glyph or 'glyphindex' glyph is selected (relatively slow operation)
Select(glyphid) **Select(glyphid, value)**	Changes the glyph's selection state. 'glyphid' may be string (glyph name), Uni (Unicode index) or integer (glyph index)
Unselect()	Deselects all glyphs in the current font (fast operation)
Message(message, question, OKstring, Cancelstring)	Shows the alert message dialog box, all the parameters but the first can be omitted
BeginProgress(title, counts)	Opens the Progress dialog box with. 'counts' - number of 'ticks'
EndProgress()	Closes the Progress dialog box
TickProgress(tick)	Updates the Progress bar, returns False if Cancel button was pressed. This is a relatively slow operation
Random(hivalue) **Random(lovalue, hivalue)**	Returns a random value (very fast operation)
ForSelected(function_name)	calls 'function_name' for each selected glyph in the current font. Function has the following format: function(Font font, Glyph glyph, glyphindex)

We will provide some examples using the functions described above when we discuss Font and Glyph classes.

Font

The Font class contains all the data that is related to the font in FontLab internal data structures. The Font class by itself has no interaction with the user interface elements. You need to use other classes such as the **fl** object of the FontLab class to interact with the font.

The most important members of the Font class are:

classes	Python list with the strings containing FontLab classes.
ot_classes	String containing OpenType classes (text that appears in the bottom control of the OpenType panel).
features	List of OpenType features. Each element of the list is an object of the Feature class.
customdata	A string that may contain any data you want to attach to a font. This data is saved in the FontLab font file so you can use this member to store information that is not editable by FontLab tools.
truetypetables	List of custom TrueType tables. Each element of the list is an object of the TrueTypeTable class.
ttinfo	TrueType information (mostly hinting-related tables).
glyphs	Array of glyphs. Each element of the array is an object of the Glyph class.

So, for example, if you want to see the advanced width of the glyph with index 12 you would write:

```
print fl.font.glyphs[12].width
```

Here “fl” is an object of the FontLab class that represents the FontLab user interface; “font” is the currently active font (object of the Font class); “glyphs” is a member of the Font class representing an array of glyphs; [12] directs us to the 12th element of the glyphs array; “width” is a member of the Glyph class representing the glyph’s advance width.

A shorter way to access the glyphs of the font is to access the Font object as an array:

```
print len(fl.font)
print fl.font[12].width
```

The first line will print the number of glyphs in the font. The second line is a shorter version of the example described above.

With the glyphs member you may perform several operations, like adding a new glyph:

```
g = Glyph()
fl.font.glyphs.append(g)
```

This example will append a new glyph to the font.

To remove all the glyphs in the font use this method:

```
fl.font.glyphs.clean()
```

Other members of the Font class represent Font Header data and you can get a list of them using the following operation:

```
print Font().__doc__
```

Two methods of the Font class relate to FontLab internal format:

Open(filename)	opens font from VFB format
Save(filename)	saves font in VFB format

These operations are not connected to the FontLab UI, so you may use them with fonts created inside a Python program and not connected to the FontLab system.

Glyph

The Glyph class represents the glyph as a data structure in FontLab. The most important members are listed in the following table:

parent	Glyph's parent object, the Font
index	Index of the glyph in the font (it is –1 if the glyph is not connected to the font)
nodes	List of nodes. Each element of the list is an object of the Node type
customdata	A string-type data that may be attached to the glyph. This data is stored to the FontLab font file (VFB) so you can use it to define glyph properties that are not supported by FontLab
note	Note defined for this glyph (string type)
mark	Color code for the glyph mark or zero if glyph is not marked
anchors	List of anchors. Each element is of Anchor type
hhints, vhints	List of horizontal or vertical hints. Each element is of Hint type
hlinks, vlinks	List of horizontal or vertical links. Each element is an object of the Link type
components	List of components. Each element is Component object
kerning	List of kerning pairs. Each element is KerningPair class object
layers_number	Number of masters in this glyph
nodes_number	Number of nodes, same as 'len(Glyph)'
width	Advance width (for the first master if glyph is Multiple Master)
height	Advance height
unicode	First Unicode index in integer form
unicodes	List of Unicode indexes
name	Glyph name
rpoint	Glyph reference point

You can get a list of operations defined for the Glyph class if you run the following line:

```
print Glyph().__doc__
```

You can replace “Glyph” with the name of any class mentioned above to get a description of its members and operations.

Here are some examples:

Marking all glyphs with a different color depending on the number of components.

Non-composite glyphs are not marked. Glyphs that have exactly 2 components are marked green and glyphs that have some other number of components or have outline and components are marked red:

```
for g in fl.font.glyphs:                  for every glyph in the current font
  c_len = len(g.components)               store the number of components (length of the components array)
  n_len = len(g)                          store the number of nodes (length of the nodes array)
  if c_len == 0:                          if number of components is 0
    g.mark = 0                            unmark the glyph
  elif c_len == 2 and n_len == 0:         if number of components is 2 and number of nodes is 0
    g.mark = 100                          Mark the glyph with green
  else:                                   if all conditions above are false
    g.mark = 1                            Mark the glyph with red
fl.UpdateFont()                           finally, update the font so the Font window will refresh
```

Selecting all glyphs that have no outline or components:

```
fl.Unselect()
for g in fl.font:
if len(g) == 0 and len(g.components) == 0:
   fl.Select(g.index)
```

First thing we do is deselect the font. It is faster to deselect all glyphs at once than to change the selection of all glyphs. In our example we are selecting only those glyphs that comply with the condition.

Please, refer to our website (www.fontlab.com) for more examples and information about macro programs. You may also examine the source code of the sample programs that are supplied with FontLab.

Index

3

3D Extrusion, 468
3D Rotate, 469

A

Actions
  Hints and Guidelines, 333
Actions, 297
Adjust Metrics, 464
Adjusting Metrics, 428
Adobe Illustrator, 281, 394
Adobe Standard Encoding, 68
AFM, 15, 141, 400, 418, 443, 445
AI, 394
Aliases Table, 347
Align, 529
AlignBottom, 527
Alignment Zones, 196, 229, 349, 487, 526, 618
  Family, 197, 490
AlignTop, 527
Alternate Substitution, 667
AMM, 445
Anchors, 229, 337, 372, 389
Anchors Panel, 343
Anisotropic Interpolation, 596, 605
ANSI, 68, 407
Append, 93
Arabic, 193
Ascender, 185, 188
Autohint, 55, 146, 460
autokerning, 15
Automatic Kerning Generation, 425
Automatic Metrics Generation, 416
Automatic TrueType Hinting, 567
Autoreplace, 461
Autoreplacing, 504
Autosave, 89
autospacing, 15, 464
axes, 588
Axis, 43
  Defining, 600
  Editing Settings, 616
  Removing, 617
axis graph, 597, 620
axis map, 597
Axis Panel, 604

B

Background, 225, 229, 323, 373
baseline, 305, 308, 411
  Property Panel, 308
Bezier curves, 54, 237
bitmap background, 323
Bitmaps, 143, 568
Bitstream, 193
blend coordinates, 593
blue marks, 236
BlueFuzz, 200, 201, 495
BlueScale, 200, 494
BlueShift, 200
BlueShift, 200, 494
Bold, 465
Break, 263, 357
BTBD, 189

C

Caps height, 185
Caret Positioning, 684
cell, 62
CFF-flavored, 128
CFF-flavoured, 632
Character, 20, 65

Character List, 473
Character mapping standard, 64
Character Set, 193
  OEM, 193
  ShiftJIS, 193
  Symbol, 193
Characters
  Deleting, 96
characters' cells, 59
class
  Anchor, 706
  Component, 706
  Feature, 704
  Font, 704
  FontLab, 702
  Glyph, 706
  Hint, 706
  Node, 706
  TrueTypeTable, 704
Class-Based Kerning, 430
classes, 106
Classes Panel, 431
Codepages, 75, 192
  Double-byte, 78
Codepages mode, 66
Collection, 100
Collection Panel, 102
Collection/Classes, 43, 270
College, 467
Compiling, 655
Component, 388
  Adding, 386
Composite, 53, 340, 385, 566
composite characters, 385
Composites, 345
connections, 239, 297, 456
Context, 668
Context Menu, 23, 82
Context-Dependent Substitutions, 668
continuous, 81, 91
Contours, 234
Control marks, 216
convert
  curve to vector, 262
Convert to
  hints, 461
  links, 461
  TrueType, 456
  Type 1, 456
Convert to instructions, 460
Converting the Kerning, 660
Coordinate Rounding, 483
copy, 90, 279
Copy characters, 90
Copying characters, 92
Copyright, 173
cubic b-splines, 237
custom TrueType tables, 55
Cyrillic, 68

D

decompose, 53, 55, 456
decomposing, 385, 387
Definition Encoding, 70
Deleting Characters, 96
Deleting Nodes, 255
Delta Instructions, 545
Descender, 188
Description, 173
design axes, 588
design coordinates, 593
Designer, 175
Directories, 23
Double Link, 539, 576
Double-Byte Codepages, 115
Drag-Drop, 49
Drop TT Hints, 461
Duplicate, 279
dynamic range, 589, 622

E

Echo, 250
Edit, 227
Edit Macro Panel, 697
Edit mode, 225, 227
Edit Window
  change a view in, 222
Editing Field, 216
Editing hints, 329
Editing Kerning, 419
EID, 170
embedded bitmaps, 55, 569

Embedding, 174
ENC, 70
Encoding, 21
encoding files
  custom, 69
Encoding Table, 67
Envelope, 373, 378, 470
EPS, 15, 394
Erase, 227
Eraser, 257
Exceptions, 670
Exchange, 318
Export Options, 139
Export Terminal, 144
Exporting Bitmaps, 573
Exporting Glyphs, 394
External Programs, 29
Extrapolation, 595, 605
extremes, 297, 457

F

FB threshold, 200
Feature Definition Language, 635
Features, 630
Features Preview Panel, 658
fina, 679
Final Delta, 551, 574
Find and Replace, 290
Font, 21
  Metrics, 398
  reencoding, 107
Font class, 704
Font Family, 21, 167
Font Features, 628
Font Formats, 128
font height, 233
Font Info, 61, 160
  Creation year, 173
Font Map Panel, 113
Font Metrics
  What are, 398
Font window, 32, 58, 60, 79, 82, 155, 216, 407
  Customizing, 155
  popup menu, 82
  selecting, 81
FontAudit, 15, 245
FontDirectionHint, 209
FontInfo
  Copyright, 173
  Notice, 173
FontInfo
  Ascender, 185
  Caps height, 185
  Created by, 173
  Descender, 185
  Description, 173
  Family Name, 165
  Font Name, 166
  Full Name, 166, 168
  Italic, 165
  Menu Name, 166
  Revision, 177
  Style Name, 166
  Subscript, 190
  Superscript, 190
  TrueType Unique ID, 178
  TrueType Version, 177
  Unique ID, 178
  Vendor Code, 178
  Version, 177
  Weight, 165
  Width, 165
  x height, 185
  XUID, 178
FontLab, 15, 128, 130
FontLab 2.0, 50
FontLab 2.5, 50
FontLab class, 702
FontLab user interface, 16, 26
Font-Level Type 1 Hints, 486
Fonts, 43
  Opening, 50
Fonts List, 472
Fonts Panel, 120
FotInfo
  Italic angle, 185
FreeHand, 281

G

GDEF, 56, 634
Generate Names, 112
Glyph, 20
Glyph class, 706
Glyph classes, 638
Glyph Geometry, 673
Glyph name, 22
Glyph Window, 34, 215
  open the, 216
glyphs, 64
  Exporting, 394
  Importing, 394
Glyphs bar, 37, 219
GPOS, 56, 634
Gradient, 469
graphical primitives, 15
green marks, 237
Grid, 309
Gridfitting, 483
GSUB, 56, 634
guideline, 315
Guidelines, 229, 310, 372
  Tracking, 314

H

hdmx, 206
head, 209
Highlight Differences, 572
Highlighting, 98
hint
  properties panel, 336
Hinting, 195, 225, 481
  character-level, 327
  Diagonals, 581
  font-level, 327
  Multiple Master Fonts, 625
  Serifs, 580
  Strategies, 574
  Symmetrical Characters, 582
  White Space, 578
Hinting Sidebearings, 565
Hints
  Type 1, 485
Hints, 327
  Editing, 329
  Tracking, 330
  TrueType, 510

I

IBM, 182
IBM Identification, 182
Identification
  IBM, 182
  Microsoft, 183
  PANOSE, 180
  PCL, 183
  Ventura Publisher, 183
Illustrator, 281, 394
Importing Glyphs, 394
Including files, 640
Index, 59
Index mode, 66
INF, 141
init, 679
Instructions
  Removing, 552
Integrating, 695
Interpolate, 373
Interpolation, 374, 511, 543, 583, 614
Intersection, 298
ISO Latin 1, 68
isol, 679
Italic angle, 185

J

Join, 263, 357

K

kern feature, 660
Kerning
  Editing, 419
Kerning, 39, 398, 399, 660
  Automatic Generation, 425
  Manual Editing, 420
  Resetting, 427
Kerning mode, 40
Key Glyph, 435

Keyboard, 27
Knife, 227, 263

L

Language, 642
Languages, 631
languagesystem, 641
License, 176
LID, 170
Ligature Substitution, 665
Ligatures, 345
Line Gap, 188
Links, 328, 532
  Editing, 331
Local Toolbar, 216
lookup, 646
lookupflag, 644
Lookups, 630
LowestRecPPem, 209

M

Mac, 187, 585
Mac Name, 168
Macintosh, 68
Macro, 59
Macro Tool, 696
Macro Toolbar, 692
Magic Wand, 227, 277, 356
Make Master, 457
Mapping Encoding, 67
Marking, 98
Mask, 229, 316
  Assigning, 319
Mask to Master, 610
master font, 585
  Designing, 608
  Selecting, 602
Masters, 43
measure, 371
medi, 679
Menu, 22
Merge, 298
Merging Fonts, 121
Meter Mode, 370
Metrics, 305
  Automatic Generation, 416
  Manual Editing, 411
  Opening Files, 443
  property panel, 308
  Saving Files, 445
Metrics Files, 400
Metrics mode, 40
Metrics window, 38, 216, 401
  Metrics mode, 410
  modes, 402
Microsoft Identification, 183
Middle Delta, 546, 574
middledot, 149
Mirror, 297, 452, 453
MM, 588
MMM, 445
mode
  Preview, 402
move, 284
Move characters, 85
Moving Nodes, 248
Multiple Master, 15, 50, 128, 328, 585
  Pinting, 154
Multiple Master Metrics, 615

N

name record, 171
Names
  Non-English, 169
  Special, 169
Names mode, 66, 68
NID, 170
node, 236
Nodes
  Deleting, 255
  Inserting, 258
  Moving, 248
  selection, 276
Non-nodes editing, 252
Note, 84, 116
Notice, 173

O

Old Style Numerals, 628
OpenType, 15, 43, 47, 50, 128, 150, 168, 169, 627

OpenType Glyph Properties, 683
OpenType Panel, 649
Operation
  Bitmap Positioning, 326
  Component Positioning, 388
  Curve, 381
  Envelope, 378
  Interpolation, 374
  Set, 382
  Set Startpoint, 384
  Transform, 282
Optical Size, 591
Optimize, 297, 304, 452, 458
OTF, 50, 632
Outline, 465
Outline Actions, 297
Output, 43
Output Panel, 657

P

Pair Positioning, 676
Panels, 42
PANOSE, 180
PCLT, 210
PFA, 47, 50, 130
PFB, 47, 50, 130
PFM, 15, 141, 400, 443, 445
PID, 170
Polygon, 391
Popup Menu
  Guideline and Hint, 332
Positioning Lookups, 672
PostScript, 15, 68, 130, 178, 394, 400
PPM, 207, 482, 526, 568
preview mode, 41
  selecting a string, 404
Preview panel, 269
  TrueType, 521
  Type 1, 504
Primitives, 391
Primitives, 43
Primitives Panel, 391
Property Panel
  Baseline, 308
  Hint, 336
  Metrics, 308
  Selection, 288
Python, 15, 17, 43, 122, 689, 690, 691, 692, 695, 696, 697, 699, 704, 705

Q

Quick Save, 418

R

Random, 470
Rearrange Contours, 384
red marks, 237
reencoding, 107
reference poiint, 382
reference point, 233, 370, 375, 706
Remove Overlap, 457
rename, 105
Reordering Features, 652
Replacement Points, 501
Reverse all, 456
Right-to-Left, 408
rotate, 284, 455
Rulers, 36, 218

S

Sample String
  Navigating, 406
save current encoding, 61
Scale, 454
Script, 642
Scripts, 631
Searching, 103
selection
  scale, 284
selection
  move, 284
  Property Panel, 288
  rotate, 284
  skew, 284
  slant, 285
Set Sidebearings, 463
Set Width, 462
Shadow, 468
Shift, 453
Show

Connection mode, 242
Control vectors, 242
Show Nodes, 242
Single Link, 534, 576
Single Positioning, 675
Single Substitution, 663
Single-Master Font, 623
Sketch mode, 225, 350
slant, 285, 311, 455
Small caps, 628
Smooth, 158
Smoothing, 207
Snap, 228, 297
Sorting, 117
splines, 54
Standard Stem Widths, 491
Standard Stems, 198, 533
Rounding, 553
Star, 391
startpoint, 236
Style, 592
Style Name, 168
Subscript, 190
Subtable Breaks, 678
supercurve, 238
Superscript, 190
Symbol, 68

T

template, 156, 229, 322
Terminal, 144
Tool
VectorPaint, 359
Tools
Eraser, 257
Meter, 225
touched points, 510
Tracking
Guidelines, 314
Tracking Hints, 330
Transform, 282
Transformation
Program, 474
Transformations, 447
TrueType, 15, 47, 50, 128, 130, 177, 178, 400, 445
TrueType curves, 237
TrueType hinting, 15
TrueType Hinting Tool, 514
TrueType Hints
Visual, 513
TrueType instructions, 146, 510
TrueType instructions, 55
TTF, 47, 50
Type 1, 15, 47, 128, 130, 177, 178, 327, 400, 445
Type 1 curves, 237
Type 1 Hinting Tool, 499
Type 1 Hints, 485
Type Manager, 178
Typo Ascender, 188
Typo Descender, 188
Typo Line Gap, 188

U

Underline, 409
Unicode, 53, 64, 71, 147, 148, 407
Codepages, 75
Consortium, 71
duplicating, 95
generate, 110
ranges, 72
remove, 112
Unicode, 110
Unicode Ranges, 72, 194
Unicode Ranges mode, 66
Unique ID, 178
UPM, 184, 232

V

Value Record, 674
VDMX, 206
VectorPaint
Contour, 362
VectorPaint, 15, 225, 359, 365
Brush, 360, 363
Contour, 360
Ellipse, 360, 368
Line, 360, 366
Pen, 360
Polygon, 360, 367

Rectangle, 360, 368
Select, 360, 361
Text, 360, 369
VectorPaint Mode, 225
Vendor Code, 178
Ventura Publisher, 183
Vertical Metrics, 348
Vertical Preview, 274
VFA, 50, 130
VFB, 47, 50
Visual Ascender, 224
Visual Descender, 224
Visual TrueType Hints, 513
VOLT, 55, 150, 685, 687, 688

W

Waterfall, 274
Weight, 590
weight vector, 594
WeightVector, 619
Width, 590
WinAscent, 188
WinDescent, 188
Windows Menu, 119
Windows Symbol Encoding, 149
Workspace, 122
www.fontlab.com, 46, 87, 127, 689, 691, 701, 707

X

x height, 185
XUID, 178

Z

zoom mode, 222